National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

NL-339 (r. 88/04)

Canada

UNIVERSITY OF ALBERTA

FINITE ELEMENT SIMULATION OF FLOW UNDER A SLUICE GATE

(C)

by

ALBERTO ALCARAZ

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND
RESEARCH IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF CIVIL ENGINEERING

EDMONTON, ALBERTA
FALL 1988

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-45633-7

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: ALBERTO ALCARAZ

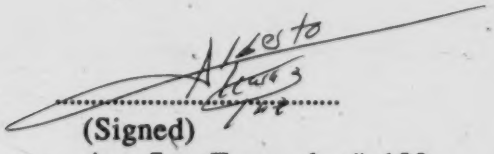
TITLE OF THESIS: FINITE ELEMENT SIMULATION OF FLOW UNDER A SLUICE GATE

DEGREE: MASTER OF SCIENCE

YEAR THIS DEGREE GRANTED: 1988

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.



(Signed)
Av. San Fernando # 193
Tlalpan 14050
Mexico City, Mexico

DATED: 16 Aug 88

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled "Finite Element Simulation of Flow under a Sluice Gate" submitted by Alberto Alcaraz in partial fulfilment of the requirements for the degree of Master of Science.

.....
Supervisor

.....

.....

Date: 26 Aug 88

ABSTRACT

The Poisson equation is used to model the flow under a sluice gate, and a method using finite elements is developed to solve the flow field. The free surface is corrected using a term computed with the Bernoulli equation. This correction term is in terms of the pressure head and the velocity head. The velocity at the free surface is obtained using a finite element model, taking advantage of the previous finite element solution of the flow.

The discharge is iterated using a Newton-Raphson technique, calculating the numerical velocity at the tip of the gate, and comparing it with the velocity prescribed by the energy level.

A discharge parameter in terms of the gate opening and the discharge per unit width is defined.

The results obtained using this model for potential flow (Laplace equation) are in good agreement with numerical studies done before. If vorticity is included, then the numerical results for the coefficients are closer to the experimental results. The effect of changing the amount of vorticity, and the thickness of the vorticity layer on the coefficients has also been studied.

ACKNOWLEDGEMENTS

The author whishes to sincerely thank Dr. P. Steffler and Dr. N. Rajaratnam for their continuous help and inspiration. Their knowledge and their ideas are reflected in this work. It would not have been possible to finish this thesis without their guidance.

Financial support for this investigation was provided by the Department of Civil Engineering and by the Natural Sciences and Engineering Research Council, their support is appreciated.

TABLE OF CONTENTS

Chapter Page

I.- INTRODUCTION 1

I.1.- LITERATURE REVIEW OF EXPERIMENTAL STUDIES 5

I.2.- BASIC EQUATIONS 12

I.3 LITERATURE REVIEW OF ANALYTICAL AND NUMERICAL STUDIES 15

II.- THE FINITE ELEMENT METHOD APPLIED TO FLOW UNDER SLUICE GATES 27

II.1.- IMPLICATIONS OF USING VORTICITY 27

II.2.- WEAK STATEMENT 32

II.3.- BOUNDARY CONDITIONS 40

II.4.- BOUNDARY VELOCITIES CALCULATIONS 42

II.5.- WALL VORTICITY COMPUTATION 45

II.6.- DISCHARGE ITERATION 50

III .- THE PROGRAM 52

III.1.- INPUT SUBROUTINE 54

III.1.1.- Subroutine PARDEF 54

III.1.1.a.- Mesh Data 54

III.1.1.b.- Flow Data 55

III.1.1- Control Data 56

III.2.- OUTPUT SUBROUTINES 58

III.2.1.- Subroutine ENC 58

III.2.2.- Subroutine OUT 59

III.2.3.- Subroutine OUTMAC 59

III.3.- MESH GENERATOR SUBROUTINES 61

III.3.1.- Subroutine SUPMAS 61
III.3.2.- Subroutine MESH 61
III.4.- GLOBAL ASSEMBLY SUBROUTINE 70
III.4.1.- Subroutine ASEMBL 70
III.5.- LOCAL ASSEMBLY SUBROUTINES 73
III.5.1.- Subroutine TRELEM 73
III.5.2.- Subroutine RECTNG 73
III.5.3.- Subroutine ELEM2 74
III.6.- BOUNDARY CONDITIONS SUBROUTINES 75
III.6.1.- Subroutine BNDRY 75
III.6.2.- Subroutine SURF 76
III.7.- SOLVER SUBROUTINES 77
III.7.1.- Subroutine UACTCL 77
III.7.2.- Subroutine SOLVE 77
III.8.- INTEGRATION POINTS SUBROUTINE 78
III.8.1.- Subroutine GAUSS 78
III.9.- NEW COORDINATES SUBROUTINE 79
III.9.1.- Subroutine NEWCOR 79
III.10.- RELAXATION FACTOR SUBROUTINE 80
III.10.1.- Subroutine RELAX 80
III.11.- VORTICITY SUBROUTINE 81
III.11.1.- Subroutine DINVOR 81
III.12.- DISCHARGE ITERATION SUBROUTINE 82
III.12.1.- Subroutine DIS 82
III.12.2.- Subroutine QCHAN 82
IV.- ANALYSIS OF RESULTS 83
IV.1.- ANALYSIS OF EXPERIMENTAL RESULTS. 83

IV.2.- POTENTIAL FLOW RESULTS.....89
IV.3 RESULTS WHEN VORTICITY IS INCLUDED. 103
V.- CONCLUSIONS.....119
LIST OF REFERENCES.....122
APPENDIX A 127

LIST OF FIGURES

Figure	Description	Page
1.	Definition Sketch.	4
2.	Boundary conditions for the sluice gate problem.	14
3.	Velocity profile for potential flow.	30
4.	Velocity profile in real flow.	31
5.	Velocity profile assumed in this study.	32
6.	Local coordinates system.	40
7.	Diagram to calculate the wall vorticity.	48
8.	Elements along the bed.	50
9.	Close up from elements along the bed.	50
10.	Flow Chart	55
11.	Supermaster elements coordinates.	64
12.	Master elements in the physical and transformed domains.	66
13.	Local coordinates for the master elements.	67
14.	Procedure to generate a grid.	69
15.	The effect of smoothing in a grid.	71
16.	Comparison between skyline and half-bandwidth schemes.	73
17.	Discharge coefficients from Rajaratnam (1977)	86
18.	Discharge parameters from Rajaratnam (1977)	88

19. Contraction coefficients from Rajaratnam (1977) 89
20. Contraction coefficients comparison between Benjamin (1956) and Rajaratnam (1977) 90
21. Downstream free surface comparison between Isaacs and this study. 92
22. Downstream free surface profile from Fangmeier-Strelkoff (1968) and this study. 94
23. Upstream free surface profile from Fangmeier-Strelkoff (1968) and this study. 95
24. Discharge coefficients from analytical, numerical and experimental studies. 96
25. Discharge parameters from analytical, numerical and experimental studies. 97
26. Contraction coefficients from analytical and numerical studies. 98
27. Contraction coefficients from analytical, numerical and experimental studies. 99
28. Streamlines for different gate opening-head ratios.(potential flow) 100
29. Meshes used to solve different gate opening-head ratios (potential flow.) 101
30. Contraction coefficients for different vorticities and for a fixed discharge. 104
31. Development of vorticity along the bed. 105
32. Effect of vorticity on the discharge coefficient. 107
33. Effect of vorticity on the discharge parameter. 108
34. Effect of vorticity on the contraction coefficient. 109

35. Discharge coefficients for different vorticities. 110
36. Discharge parameter for different vorticities. 111
37. Contraction coefficients for different vorticities. 112
38. Meshes used to solve different gate opening-head ratios. (vorticity included) 113
39. Streamlines for different gate opening-head ratios and a prescribed vorticity. 114
40. Effect of the thickness of the bottom layer on the contraction coefficient for a fixed discharge. 116
41. Effect of the thickness of the bottom layer on the discharge coefficient. 117
42. Effect of the thickness of the bottom layer on the discharge parameter. 118
43. Effect of the thickness of the bottom layer on the contraction coefficient. 119
44. Comparison of numerical downstream free surface profiles with experimental results. 114

NOMENCLATURE

a	gate opening
C_c	contraction coefficient
C_d	discharge coefficient
C_{d2}	discharge parameter
f_i, v_i	shape and basis functions
g	acceleration of gravity
H	total energy level
J	Jacobian
$\overline{n}$	vector normal to the boundary
P/γ	pressure head
q	discharge per unit width
Q	volume rate flow
r, s	local coordinates
u	velocity in the "x" direction
v	velocity in the "y" direction
V	average velocity
x, y	global coordinates
y_o	upstream depth
y_d	downstream depth
α	relaxation factor
∇	Laplace operator
η	vorticity
ψ	stream function
Ψ_j	nodal values of the stream function

I.- INTRODUCTION

A control is defined as any channel feature, natural or man-made, which fixes a relationship between depth and discharge in its neighborhood. Sharp crested weirs, overflow spillways, free overfall structures and underflow gates fall into the class of man-made features.

It is important to know the functioning of the control itself and what is the extent of its influence on the flow. Thus it is necessary to know the discharge and the free surface profile for given flow conditions.

In the case of underflow gates, an appropriate analysis of the gravity flow is needed with emphasis on the determination of the free surface profile and the discharge for a given total head and a gate opening.

According to Chow (1959) underflow gates may be classified as: 1) vertical sluice gates, 2) radial or Tainter gates and 3) rolling gates. These structures may be used as controls at the crest of an overflow spillway, or at the outlet from a lake to a river or irrigation canal etc. In order to design these structures an accurate analysis has to be done. This problem has been studied by a number of investigators since the beginning of the century. There has been a logical development; in the beginning, experimental studies were made, basically because these structures were being built and it was necessary to know if they were appropriate; later some analytical studies involving complex variable theory were done. When digital

computers were introduced, the numerical analysis became very popular. Then after these studies were done some people went back to do laboratory work in order to verify the numerical and analytical results. At the present time, we are again in the stage of computational studies but more laboratory results are needed since as it has been found, the problem of gravity flows is very complex and every time new numerical work is done, new questions arise.

Until now the analytical and the numerical studies had always something in common, the equation that was used to model the flow was Laplace equation, implying potential flow theory. All the discrepancies between the experimental results and the analytical or numerical ones were said to be because of the boundary layer on the bed. It was not until recently that this assumption was challenged; again because the laboratory experiments showed that it could not be the only reason for these discrepancies. This study has a new approach and it is that the irrotational characteristic of the flow is modified, so the equation that is used to model the flow field is Poisson equation.

The numerical studies can be divided basically into: finite differences, finite elements and boundary elements models. In recent years both finite differences and finite elements have been used but there are more references related to finite element models. As will be mentioned in chapter I, the finite elements models that have been used involve tremendous amount of computer time and this is because a great number of degrees of freedom are needed and since the problem is non linear an iterative scheme has to be used.

The aim of this study was to develop a model that could be used to solve the sluice gate problem but could also be easily extended to solve different flow problems. The idea was to try to reduce the computational effort involved so that the model could be transferred from a main frame into a microcomputer. In order to achieve this, the model should give good results with fewer degrees of freedom than what has been commonly used. The numerical technique that was used was the finite element and this technique was preferred over the finite differences because of its ease to model complex geometries and also because once a mathematical problem is solved, it can be applied to different physical problems just by imposing the corresponding boundary conditions.

This work is divided into 4 chapters, the first one presents a literature review that helps us to understand the development of the different solutions obtained and how they were derived; it also presents the basic equations and the mathematical formulation. The second chapter presents the numerical implementation of the problem; it also serves as an introduction to chapter III which explains the computer program. Finally chapter IV deals with all the numerical results and also with the analysis of some experimental results obtained by Rajaratnam in a previous study (1977).

Figure 1 shows the definition sketch used throughout this study.

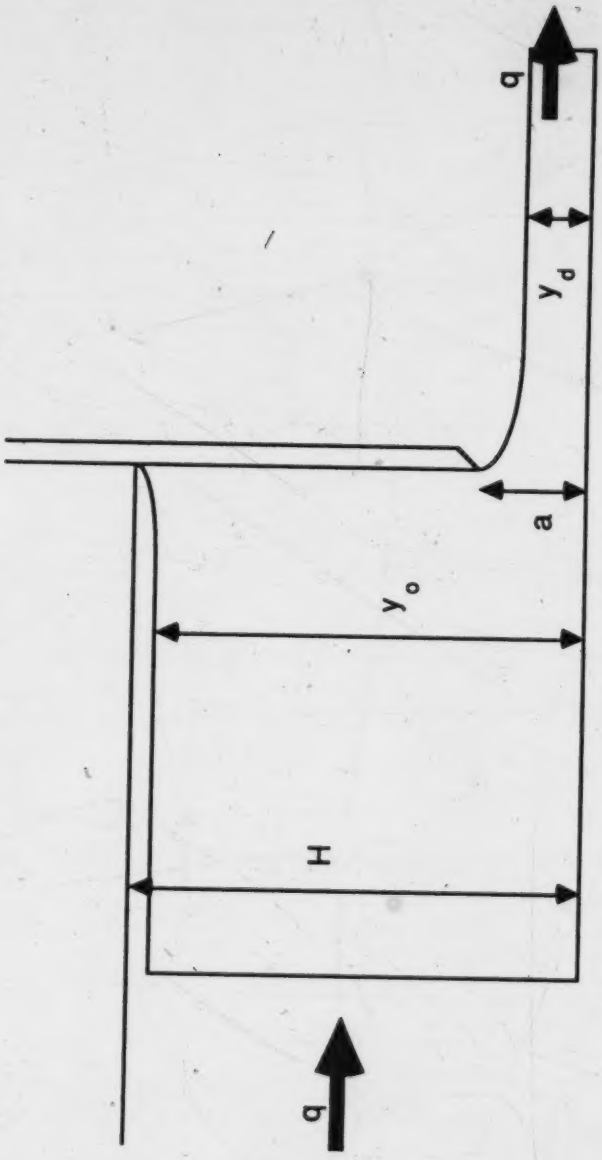


Fig. 1.- Definition sketch.

I.1.- LITERATURE REVIEW OF EXPERIMENTAL STUDIES

The experimental analysis of flow under sluice-gates goes back to the first quarter of this century, when studies were made for the construction of the Assuan Dam. Hurst and Fraser (1923-1924) reported some of the results that were obtained after more than 10 years of work. Their aim was mainly to describe the relations between models and prototypes and hence their results were presented as the differences found between what was measured in the models and in the real sluice-gates. In the theoretical discussion included in their paper, they mention that the discharge can be represented by the equation

$$Q = C_d A \sqrt{2g(H - F)}$$

where C_d is a coefficient independent of the head but varying with the sluice opening. F is a constant depending on the opening, A is the area of the opening and H is the head above the floor of the intake. It can be seen that this equation is similar to the one used at the present time although now the term (H - F) is replaced by the upstream depth.

After this work, Addison (1937-1938) did more experiments also related to the construction of the Assuan Dam. The possible disposition of sluice-gates and the conditions of flow were classified as:

1) Free flow through high-head sluices (water discharging freely into air on the downstream side).

2) Submerged flow through high-head sluices (downstream water level above sluice openings).

3) Free-flow through low-head sluices.

4) Submerged flow through low-head sluices.

Addison made a series of experiments using a 1/25 scale model of the Assiut Barrage and a 3/100 scale model of the Assuan type "A" sluice. He noted that the formula proposed by Hurst was in general applicable but that the proportions of the sluice and sluiceways had a marked effect on the values of the coefficients C_d and F. The value that he obtained for C_d (discharge coefficient) was constant for some of his experiments with a value of 0.615, but for some others the value was between 0.671 and 0.729 as the sluice gate opening increased. Addison concluded that there was not a very clear correlation between the value of the ratio F/a and the gate opening or the shape of the opening. In this study, the value of "F" was taken as the depth at the vena contracta, while in Hurst's study (1923-24) this value was not determined. With that in mind F/a is the contraction coefficient. He also concluded that under ideal conditions the values of the contraction coefficient and the discharge coefficient would be numerically equal.

Henry (1950) in his discussion of a paper written by Albertson et al. on Submerged Jets, reported some results on free discharge under a sluice gate. He mentioned that at lower degrees of

submergence, (obviously free discharge would fall under this category, no submergence at all), gravity has an impact on the free surface and then the Froude number should be included in the analysis. He published a graph showing y_o/a versus a discharge coefficient C_d defined as:

$$C_d = \frac{F_B}{\sqrt{\frac{2y_o}{a}}} \qquad \text{with} \qquad F_B = \frac{q}{a\sqrt{ga}}$$

being "q" the discharge per unit width. Substituting F_B into the equation for C_d we obtain:

$$C_d = \frac{q}{a\sqrt{ga}} \frac{1}{\sqrt{\frac{2y_o}{a}}}$$

and

$$C_d = \frac{q}{a\sqrt{2gy_o}}$$

so in fact C_d is the discharge coefficient defined in the way that it has been used since then.

According to his results, the discharge coefficient is asymptotic to 0.6 for high ratios of y_o/a (gate closing) and for y_o/a equal to 2, C_d is 0.5. He also performed an approximate theoretical analysis assuming hydrostatic pressure distribution throughout the flow, no energy loss and uniform velocity upstream and downstream of the gate to verify the trend of the curves that he obtained experimentally.

Benjamin (1956) published a paper discussing the effects of placing rigid obstacles in a channel and a sluice gate falls into the category of an obstacle changing the flow from a subcritical stream to a supercritical one. He developed a general method for calculating the form of the receding stream when waves were absent and presented an example using the sluice gate problem. The treatment that he followed was based on ideal fluid theory and later he did some experiments to test his theoretical results. He combined two different methods, one was used to calculate the form of the converging stream which issues from under the gate and the other to calculate the flow in the immediate neighborhood of the gate. He found that there is a variation of the contraction ratio with the Froude number and established a relation between the sluice opening and the total discharge.

In his experiments Benjamin observed that the thickness of the boundary layer was about one-fifteenth of the sluice opening and concluded that the frictional effects were far more significant than the final error created by using the ideal fluid flow theory, and that the discrepancies observed with real flow were due to friction alone. He did not report any results related to the discharge coefficient but he presented a graph showing a/y_o versus the contraction coefficient . He did two series of experiments with two different gate openings and varying y_o. For both experiments the results are above the theoretical curve. It seems that for small values of a/y_o (less than 0.1), the three curves converge but not for the rest. The curve for a small gate opening lies above the one with a larger gate opening. As mentioned before he assumed that the discharge under

the sluice was affected by friction only and that the effective gate opening should then be reduced to a-δ where δ was considered to be roughly equal to the thickness of the boundary layer. The depth downstream was then also reduced by the same δ so the discrepancy in the contraction ratio was approximately proportional to δ/a. He applied this concept to the two series of experiments that he had done and he obtained quite a good agreement but he did not pursue this further, simply suggesting that further experimental work was necessary especially on the boundary layer.

Rajaratnam and Subramanya (1967) published a paper where they explained how they developed an experimental curve for the discharge coefficient which could be used for both free and submerged flows. They also compiled the results from various investigators for the contraction coefficient which was found to be a function only of a/y_o when the viscous effects were neglected. By applying the energy equation between a section before the gate and a section after, they derived an equation for the unit discharge:

$$q = \frac{1}{\sqrt{\alpha - \left(\frac{y_d}{y_o}\right)^2}}\, y_o \sqrt{2g\Delta H}$$

where α is a kinetic energy correction factor. Furthermore, they reduced the above equation to:

$$q = C_d a \sqrt{2g\Delta H}$$

with

$$\Delta H = H - y_d$$

and

$$C_d = \frac{C_c}{\sqrt{\alpha - C_c^2 \left(\frac{a}{y_o}\right)^2}}$$

which is valid for both free and submerged flows.

Based on their experiments, they found that an average value for α was 1.08 in the case of submerged flows and was close to the unity for the free discharge. They approximated the value of α to 1 and they also took a constant value for C_c of 0.61, so finally their C_d was:

$$C_d = \frac{0.61}{\sqrt{1 - (0.61)^2 \left(\frac{a}{y_o}\right)^2}} \quad \text{and} \quad C_d = f\left[\frac{a}{y_o}\right]$$

They also worked with the momentum equation in order to help to solve the submerged flow problems and presented a graph to determine the nature of the flow (free or submerged).

Rajaratnam (1977) published another paper related to the flow immediately below sluice gates but this research was focussed more on the contraction coefficient and on the free surface profile than on the discharge coefficient. He found that the profiles of the different experiments could be described by one dimensionless curve. To do this he defined two length scales, one for the "y" direction and

another for the "x" direction. The length scale for Y was equal to ($a-y_d$) and for X was the value of x in which (a-y) was equal to 0.75Y; the X scale was chosen after some trials. It was noticed that the numerical values for the contraction coefficient found by several investigators differed from the experimental ones by as much as 14% in some cases. After the comments by Benjamin he decided to work with the boundary layer and for some of his experiments he calculated the displacement thickness and found that the corrections were rather minor and concluded that they did not account for the large observed differences between the theoretical and experimental values of C_c.

I.2.- BASIC EQUATIONS

The flow under sluice gates has always been considered as being steady, two dimensional, incompressible and irrotational, and hence it can be treated with the Potential Flow theory.

In this study the irrotational characteristic is not considered, so it is like having a "rotational potential flow", the governing equation for this type of flow is the Poisson equation (for potential flow the governing equation is Laplace equation) which for the stream function Ψ is:

$$\frac{\partial^2\Psi}{\partial x^2} + \frac{\partial^2\Psi}{\partial y^2} = -\eta$$

where η is the value for the vorticity, which is defined as:

$$\eta = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$$

If potential flow is to be considered then this value should be made equal to zero.

The boundary conditions for this problem can be expressed as Dirichlet (essential) or Neumann (natural) boundary conditions. In this work the Dirichlet type conditions were applied to the free surface. This type of boundary conditions implies that the value for the stream function Ψ has to be specified. For the boundary where the flow is uniform the Neumann type were applied, this means that

the normal derivative of the stream function has to be specified. The boundary conditions applied in this study are shown in figure 2.

Since the free surface is a streamline, we may write the Bernoulli equation:

$$\frac{P}{\gamma} + Y + \frac{V^2}{2g} = H \qquad 1.1.1$$

or

$$\frac{P}{\gamma} = H - Y - \frac{V^2}{2g} \qquad 1.1.2$$

and

$$V = \frac{q}{Y} \qquad 1.1.3$$

so

$$\frac{P}{\gamma} = H - Y - \frac{q^2}{2gY} \qquad 1.1.4$$

where (P/γ) is the pressure head, H is the total head, Y is the depth, q the discharge per unit width and g is the acceleration of gravity. This equation will be used to compute the correction term for the free surface.

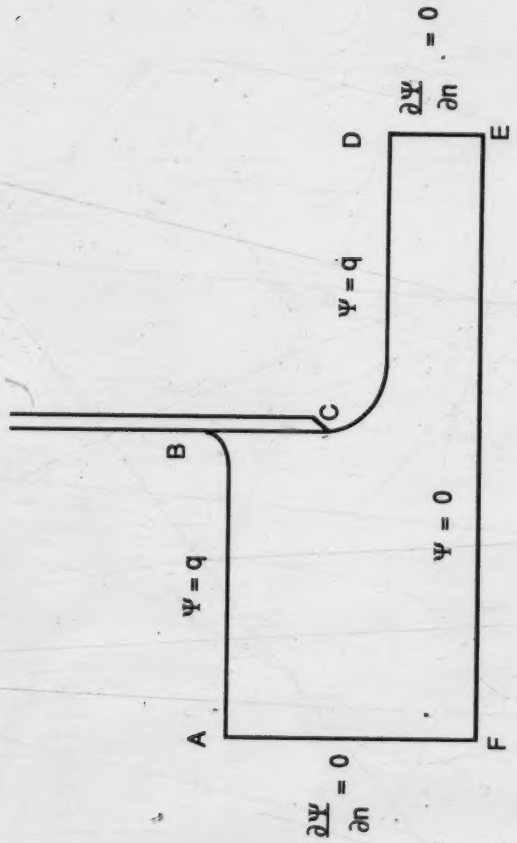


Fig 2.- Boundary conditions for the sluice gate problem.

I.3 LITERATURE REVIEW OF ANALYTICAL AND NUMERICAL STUDIES

The numerical methods to solve engineering problems go back to the 1940's. Southwell and Vaisey (1940) published "Relaxation Methods in Engineering Science". After that Southwell (1946) published "Relaxation Methods in Theoretical Physics", and there he made the distinction between 'orthodox' methods and 'relaxation' methods: "for both, the point of departure is a physical or engineering problem formulated in mathematical terms ..... faced with this mathematical problem 'orthodox' analysis applies to it the methods of pure mathematics; they take no account of the inescapable uncertainty of physical data, they do not in fact achieve the wanted generality, being restricted (usually) to boundaries of simple geometrical shape. For 'relaxation' methods, on the contrary, the boundary shape has small importance; a problem solved for one could be solved for any. A price is paid in theoretical precision, firstly because they substitute for the specified governing equation an approximation in which its differentials are replaced by finite differences, secondly because the approximation is not solved exactly. The principle which differentiates the relaxational approach is that it is implemented by a technique which fixes attention not on the wanted function but on the data of the problem, seeking to leave unaccounted parts so small as to be comparable with errors of observation. Thus stated this principle seems to have been first applied in the "Moment Distribution Method" of Hardy Cross (1924)."

This is similar to a Finite Differences approach; in his book Southwell defined a "net" and "nodes" where the function that it is being looked for will be calculated. A set of simultaneous algebraic equations was obtained and then instead of attacking it by analytical methods they studied the effects of standard "operations" upon "residual forces" which constituted errors. These ones had to be "liquidated". Obviously their methods involved a lot of work since the use of computers was very limited, instead a lot of graphical work was done. The relaxation procedure itself was dependent on the problem, understanding relaxation as the imposition of displacements. Relaxation Methods were applied to free surface flows and he mentioned that "free stream-line problems are among the hardest (from a purely computational standpoint) that have yet been confronted, and no routine procedure can be guaranteed to solve every example."

Strelkoff (1964) proposed an analytical method to solve 'highly curvilinear gravity flows'. His approach involved the iterative numerical solution of an integral equation derived by conformal mapping and singularity distribution. The method was applied to the flow over a sharp crested weir. He considered a potential flow and the integral equation described the influence of the flow profile of the three primary variables: gravitational attraction, discharge and channel geometry.

Klassen (1967) proposed a solution for the sluice gate problem using an integral equation. The purpose of his method was to find the free streamline and all other properties of the flow knowing the total

head, the discharge and the gate opening; so it was meant just to calculate the contraction coefficient for a given flow.

Fangmeier and Strelkoff (1968) applied Strelkoff's method to the flow under sluice gate. Their aim was to develop a general program to solve the sluice gate problem that could be executed by a digital computer. The output would consist of the flow profile and discharge coefficient for a given ratio of gate opening to total head, which was the only data describing the flow that was supplied to the computer. They worked with four different planes: the physical plane (Z), the complex potential plane (ω), an auxiliary plane (ξ), and the circle plane (T). The sluice gate problem was transformed into an equivalent flow about a flat plate and a solution was derived by transforming that plate into a unit circle and applying the Milne-Thomson circle theorem. After all the transformations, an integral equation in terms of an unknown discharge and an unknown distribution of elevation as a function of velocity potential resulted. Then that equation was approximated by a non-linear ordinary differential equation which was solved by an iterative, numerical step method.

The way that the discharge "q" was obtained is interesting: they had to guess a first "q" and then solve the differential equation. When a solution was obtained and the correct "q" was found, the value of the real part part of the complex variable T had to be zero. It was noted that if "q" was too small, that value would be greater than zero, and if "q" was too large that value would be smaller than zero, so after each iteration they monitored the value of "r" and as soon as they noticed a tendency, the discharge was adjusted and the

solution restarted from initial conditions. As the correct value of "q" was approached, the solution behaved properly over a greater region of integration.

Their program was coded in FORTRAN and run on an IBM-7040. The minimum computer time needed was 75 minutes. They compared their results with the experimental results from Benjamin (1956) and found that their contraction coefficients were much lower, they were even lower than those by Pajer (1937), by Perry (as quoted by Benjamin), and by Southwell (1946). Their discharge coefficients were in very good agreement with the experimental ones by Henry (1950).

Larock (1969) introduced another method; the way the flow was treated was as if it were an extension of a related gravity-free flow. He neglected the effects of the upstream free surface and gravity, so he considered the upstream free surface as a horizontal line. He applied the complex function theory using the circular-arc hodograph plane. His goals were to obtain a solution for the gravity-affected flow from planar sluice gates, to develop a solution which would require only a modest amount of computer time and that would be applied for any gate inclination. The last objective was the one that made a big difference. He began with a "reference depth ratio" (y_o / y_d) and then for that ratio found the gate opening. His method was also an iterative method. In the first iteration gravity free flow was considered so a first estimate of the free surface could be obtained, then he computed a gravity term and adjusted the angle of inclination of the gate. The process was continued until a tolerance was met. Usually three iterations were needed. The rapidity of

convergence was apparently due to two causes: the pressure on the free surface was imposed and then the free surface was adjusted in order to satisfy the other conditions; since the flow was described by Laplace equation (elliptic) any change at one point on the boundary would affect all the solution. In the same way as Klassen (1967), the discharge was prescribed so his method was appropriate to solve a particular flow. His results for C_c were in good agreement with those of Fangmeier and Strelkoff (1968) but also were lower than the experimental ones. He did not report anything about the discharge coefficient because he used prescribed discharges.

The first finite element analysis for the flow under a sluice gate was published by McCorquadale and Li (1971). They noted the restrictions of the analytical solutions, especially those concerned with the geometry of the gate and the consideration of having a horizontal upstream free surface. The way that they proposed to solve this problem was similar to that of Southwell and Vaisey (1946) but instead of using finite differences they used finite elements. A variational principle proposed by Luke (1967) was used. After applying the finite element theory it yielded a set of linear algebraic simultaneous equations which were solved by the successive over-relaxation procedure. In the first guess, they considered a horizontal upstream free surface and for the downstream free surface an elliptical curve was selected to describe the outflow, then they calculated an error estimate using the computed and prescribed energies to adjust the free surface. Then a new value for the discharge was computed from the energy equation. The adjustment of the free surface was determined by the values of

the constants that defined the ellipse. These values were chosen such that they minimized the error in energy. To do this different combinations of the constants were tried, computed and plotted the error. This method involved a lot of graphical work in order to determine these optimal values of the constants.

Chung (1972) used conformal mapping, analytical continuation and perturbation methods combined to obtain the solution for the sluice gate problem correctly up to the second order. He mapped the physical plane to the complex plane and this one to the T plane (unit circle). The expressions that he obtained for the discharge coefficient and the contraction coefficient were:

$$C_d = 0.61102\ (1 - 0.41152\ a/H + 0.30311(a/H)^2 + \ldots\ldots)$$

$$C_c = 0.61102\ (1 - 0.10601\ a/y_o + 0.09833(a/y_o)^2 + \ldots\ldots)$$

He considered the upstream free surface as being horizontal. His results for C_c are in very good agreement with those of Fangmeier and Strelkoff (1964), but for the downstream free surface profile the results differ a little bit, Chung obtained a lower profile. He also applied his method to inclined sluice gates and derived expressions for different angles of inclination. Finally he mentioned that the method could be applied to radial sluice gates.

Chan, Larock and Herrmann (1973) developed a general method for free surface ideal fluid flows using the finite element technique. The properties that they looked for in the method were: it should be able to analyze either confined or free surface flows in either the presence or absence of gravitational effects; the method should be versatile so that the problem involving complicated solid boundaries could be analyzed directly; and the method should use a

rational algorithm for calculating the free surface coordinates. They developed a variational principle which mathematically is equivalent to the Laplace equation subject to the natural boundary conditions (the value of the derivative of the function respect to the normal direction is specified on the boundary), the principle has two terms, physically the first term represents the total kinetic energy of the moving fluid in the domain and the second term is related to the amount of work done by an impulsive pressure in starting the motion from rest. They divided the flow domain into linear elements. The system of equations obtained was solved by Gaussian elimination. In the problems that they solved, the "reference velocity" was known, so the discharge was specified.

The importance of this work is that for the first time a numerical algorithm to calculate the free surface was developed. In their algorithm they assumed a free surface and imposed constant pressure, then solved the flow using the finite element method. Obviously the velocity condition was not satisfied so from the finite element solution they calculated the values of the velocity components for each node and used a curved fitting scheme to find a new free surface that would satisfy the velocity condition (Bernoulli equation). With that "improved" free surface they would solve again. The procedure was repeated until a certain tolerance was met. One of the examples that they solved was flow from a 45° slot, that could be compared with the flow from a 45° sluice gate if the upstream free surface is considered horizontal. They compared their results with those from Larock (1969) and found that they were in good agreement. For problems involving a free surface, 10 to 20 iterations

were normally required. It is important to mention that the only nodes that were adjusted were those on the free surface, the rest remained in their original position.

Diersch, Schirmer and Busch (1977) developed a general finite element model based on the Chan et. al. method (1973). They used the same variational principle and the same algorithm to calculate the free surface but they made it general, so it could be used in a wider range of applications. One significant feature was that they calculated the discharge which was not prescribed in Chan's work. Another interesting feature was the use of "flexible finite element networks"; this means that the position for all the nodes was recalculated after each iteration, thus making sure that the net was logical in the sense that there was no overlapping of elements. The movement of the net is specially difficult in some cases, for example in the flow over a spillway because the floor is not horizontal, so they developed expressions to correct in both "x" and "y" directions (Chan et al. (1973) just corrected in the "y" direction). The way to iterate the discharge was dependent on the location of the free surface using the Bernoulli equation and continuity between the far upstream and far downstream sections. For the sluice gate problem they usually required 17 to 25 iterations using 1061 nodes. Results concerning the discharge coefficient only were reported, which were very close to Henry's (1950) experimental results and are supposed to be more accurate than the ones by Fangmeier and Strelkoff (1968). It is important to note the large number of degrees of freedom that were needed in order to get good accuracy.

Isaacs (1977) did some work with finite elements but he used a new approach. He solved directly the Laplace equation and did not use any variational principle. Also the element that he used was different. He divided the flow domain in cubic triangular finite elements, with the values of ψ and its first derivatives as nodal parameters at the apex nodes and the value of ψ as a nodal parameter at the centroid. The algorithm that he used is: first he assumed a discharge "q" and computed the far upstream and far downstream depths from the energy equation, then as a boundary condition downstream he calculated an estimate of $\frac{\partial \psi}{\partial n}$ as $v=\sqrt{2g(H-y)}$,he solved the flow field using the finite element method and checked the other condition (ψ = q), in general it was not the same so the free surface was altered according to:

$$\delta n = \frac{q - y}{\frac{\partial \psi}{\partial n}}$$

and the flow was solved until convergence was met except at the tip of the gate, convergence here occurs just for the right "q" since the energy level there is controlled by the discharge, the program was controlled interactively by the user and he would decide when to input a new trial for "q". Isaacs did not develop a method to compute the correct discharge. It depended on the user's ability to correct the initial discharge. He tried the method for a gate opening-head ratio of 0.3 and the results that he obtained were within 0.001 of the ones by Fangmeier and Strelkoff (1968) and they were also way below from

the experimental ones. He studied the boundary layer using the Von Karman-Pollhausen method and also concluded that it could not explain the difference between experimental and numerical results for C_c . His discharge coefficient was also in good agreement with previous analysis. The problem with this method was essentially that it was difficult to get the discharge. The number of degrees of freedom were less than the number used by Diersch et. al.

Masliyah, Nandakumar, Hemphill and Fung (1985) reexamined the finite difference scheme because of its conceptual simplicity and computational ease in implementation. They developed a method that involved body-fitted coordinates to treat the free surface, overcoming one of the drawbacks of the finite differences methods. The grid is generated as a system produced by the solution of a pair of elliptic differential equations. The method has the ability of placing a large number of grid points easily around the gate opening. The coordinate lines are not orthogonal. The boundary conditions for the numerical implementation are in terms of the unknown discharge (Q) and the unknown free surface location (h); an assumption for these unknowns is made, then the stream function values around the boundary are specified, a suitable grid is generated and the Laplace equation is solved over the domain. The boundary condition at the tip of the gate is used with the interior solution to update the value of Q. The Bernoulli equation along the free surface is used to improve the location of the free surface. In the reported results, they mentioned that if the grid is changed from 11 x 27 to 15 x 37 nodes the difference in the discharge coefficient is just 0.2%. Their results are in good agreement with the ones by

Fangmeier and Strelkoff (1968) but deviate more from the experimental results than the ones by Diersch et al. (1977).

Heng, Mitsoulis and Prinos (1986) used finite elements with a different iterative technique and boundary condition specification. They combined the natural and the essential boundary conditions; upstream of the gate they specified essential boundary conditions (ψ is specified on the boundary) and downstream of the gate natural boundary conditions ($\frac{\partial \psi}{\partial n}$ specified). To begin a discharge q was assumed and the far downstream depth located using the Bernoulli equation, then they assumed a free surface between the far downstream and the tip of the gate, the upstream surface was taken horizontally for the first iteration. The Laplace equation was solved after applying the proper boundary conditions, the constant pressure condition on the free surface generally would not be satisfied so the pressure there was computed using again Bernoulli. The free surface was updated with this results along with the velocity obtained from the finite element solution and the process repeated until convergence for that discharge. Usually 6 to 10 iterations were required to obtain a free surface with relative changes of less than 10^{-4} of the downstream depth. Once convergence was obtained the value of the numerical velocity at the tip of the gate was checked against the fixed value which is dictated by the energy level. If they were not within certain tolerance a new discharge was computed using a Newton-Raphson scheme. Since this is the algorithm used in this study, it is further described in the next chapter. They found that between 3 and 5 iterations for the discharge were needed. The

elements that were used in their study were quadratic triangles. They found that with their method the discharge and contraction coefficients (C_d and C_c) were very sensitive to the density of the grid. When they used 975 nodes their results for C_d were different from the experimental ones by Henry (1950) in about 4% but when 1455 nodes were used they were in very good agreement.

II.- THE FINITE ELEMENT METHOD APPLIED TO FLOW UNDER SLUICE GATES

II.1.- IMPLICATIONS OF USING VORTICITY

In all the studies that were consulted, the mathematical modelling was done using the Laplace equation; that is considering potential flow. When potential flow is considered, the depth determined upstream of the gate (when the flow is uniform) is the subcritical depth for that discharge and downstream of the gate the depth is the supercritical one. These results give lower contraction coefficients compared with experimental measurements. The boundary layer was first considered to be an important factor for this but as Rajaratnam (1977) showed, the consideration of the displacement thickness does not make a significant difference when it is added to the depth obtained numerically.

In this study the irrotational characteristic of the flow is taken into account. For potential flow the velocity profile for a given section is uniform (fig. 3). In reality the velocity profile is like the one shown in figure 4. The velocity profile assumed in this study is shown in figure 5.

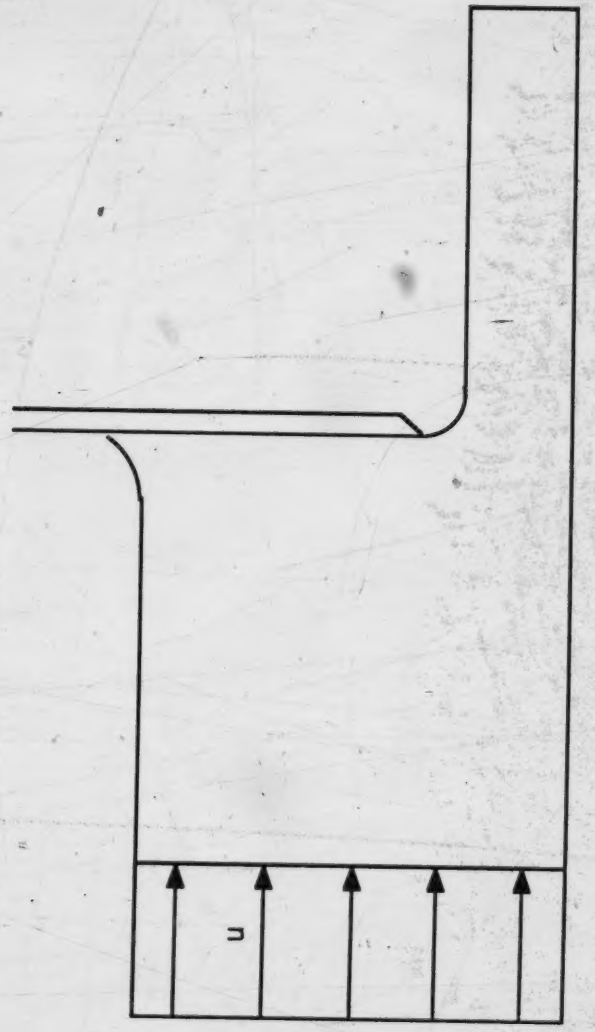


Fig. 3.- Velocity Profile in Potential Flow.

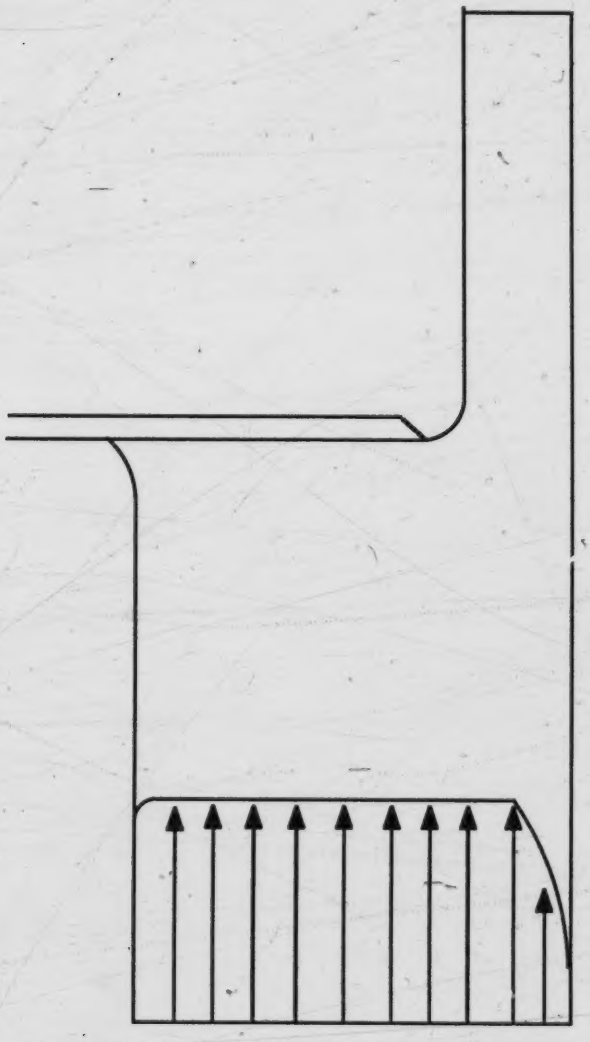


Fig. 4.- Velocity profile in a real flow.

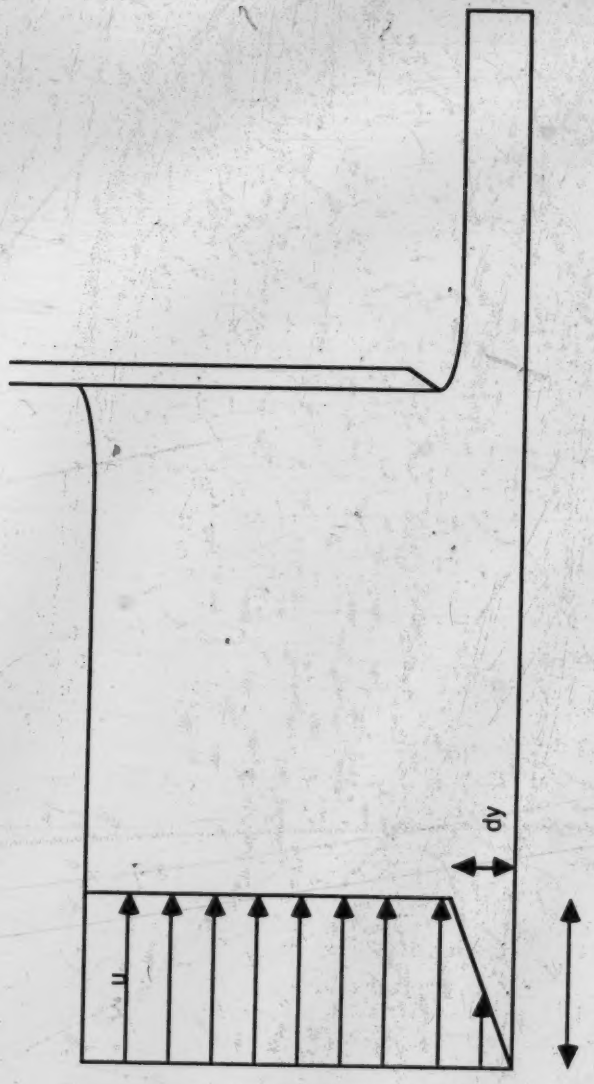


Fig.5.- Velocity profile assumed in this study.

The vorticity function is defined as:

$$\eta = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \qquad 2.2.1$$

With potential flow, by definition:

$$\eta = 0$$

Near the (horizontal) bed of the channel we may assume that v=0 and the velocity profile is assumed to be like the one in figure 5 then:

$$\frac{\partial v}{\partial x} = 0 \qquad \text{and} \qquad \eta = -\frac{\partial u}{\partial y} = -\frac{\partial^2 \psi}{\partial y^2}$$

To start the solution potential flow is calculated. After each iteration the wall vorticity can be estimated for the elements along the bed. The stream function distribution is then resolved. When a final solution for the flow field is obtained, the vorticity would also have been converged to a certain value. In section 2.5 numerical considerations and wall vorticity estimation are discussed.

In the following section the finite element method will be applied to the Poisson equation in order to develop a numerical algorithm to solve the flow field.

II.2.- WEAK STATEMENT

The equation to be solved is, as mentioned earlier:

$$\frac{\partial^2\Psi}{\partial x^2} + \frac{\partial^2\Psi}{\partial y^2} = -\eta \qquad 2.3.1$$

subject to the boundary conditions shown in figure 2.

The Finite Element theory is explained in several standard textbooks (Martin (1973), Stasa (1985, Zienkiewicz (1977)). First multiply equation 2.3.1 by a set of test functions, v_i, and integrate over the domain.

$$\int_\Omega v_i\left(\frac{\partial}{\partial x}\left(\frac{\partial\Psi}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\partial\Psi}{\partial y}\right)\right) d\Omega = -\int_\Omega v_i\eta \; d\Omega \qquad 2.3.2$$

The divergence theorem states that:

$$\int_\Omega \overline{\nabla}\bullet\overline{V}\, d\Omega = \int_\Gamma \overline{\nabla}\bullet d\overline{n} \qquad 2.3.3$$

where

$$\overline{\nabla} = \left(\frac{\partial}{\partial x^i} + \frac{\partial}{\partial y^j}\right)$$

and the Green-Gauss theorem states:

$$\int_\Omega \beta(\overline{\nabla}\bullet\overline{w})\, d\Omega = \int_\Gamma \beta\overline{w\, n}\, d\Gamma - \int_\Omega \overline{\nabla}\beta\bullet\overline{w} d\Omega \qquad 2.3.4$$

β is a scalar function.

Now, if it is considered that:

$$v_i = \beta$$

and

$$\overline{w} = \left(\frac{\partial \Psi}{\partial x} i + \frac{\partial \Psi}{\partial y} j\right)$$

then the Poisson equation may be written as:

$$\int_{\Omega} v_i(\overline{\nabla} \bullet \overline{w}) d\Omega = - \int_{\Omega} v_i \eta \, d\Omega \qquad 2.3.5$$

Applying these theorems equation 2.3.2 becomes the weak statement for the Poisson equation (2.3.6):

$$\int_{\Omega}\left(\frac{\partial v_i}{\partial x}\frac{\partial \Psi}{\partial x} + \frac{\partial v_i}{\partial y}\frac{\partial \Psi}{\partial y}\right) d\Omega \; - \int_{\Gamma} v_i\left(\frac{\partial \Psi}{\partial x} n_x + \frac{\partial \Psi}{\partial y} n_y\right) d\Gamma \; = \; - \int_{\Omega} v_i \eta \, d\Omega$$

In the finite element method the stream function is approximated by:

$$\psi \cong \sum_{j=1}^{N} f_j(x,y) \Psi_j \qquad 2.3.7$$

where f_j are a set of functions called "shape functions". The Ψ_j are values of ψ at the nodes. If the above equation is substituted into the weak statement the following equation is obtained:

$$\int_{\Omega}\left(\frac{\partial v_i}{\partial x}\frac{\partial f_j}{\partial x}\Psi_j + \frac{\partial v_i}{\partial y}\frac{\partial f_j}{\partial y}\Psi_j\right)d\Omega - \int_{\Gamma} v_i \frac{\partial f_j}{\partial n}\Psi_j \, d\Gamma = - \int_{\Omega} \eta \; v_i \, d\Omega$$

The second term represents the natural boundary conditions. In this case we set the shape functions for the essential boundary conditions ($\psi = 0$, $\psi = q$, and $\frac{\partial \psi}{\partial n} = 0$) as v_i=0, so this term vanishes for the rest of the boundary. After a solution for the flow field is obtained, this term will be used again to compute the boundary velocities. After eliminating this term we obtain:

$$\int_{\Omega}\left(\frac{\partial v_i}{\partial x}\frac{\partial f_j}{\partial x}\Psi_j + \frac{\partial v_i}{\partial y}\frac{\partial f_j}{\partial x}\Psi_j\right)d\Omega = - \int_{\Omega} \eta \; v_i \, d\Omega \qquad 2.3.8$$

Since the Ψ_j are not functions of "x" and "y", the last equation can be written as:

$$\int_{\Omega}\left(\frac{\partial v_i}{\partial x}\frac{\partial f_j}{\partial x} + \frac{\partial v_i}{\partial y}\frac{\partial f_j}{\partial y}\right) d\Omega \; \Psi_j = - \int_{\Omega} \eta \; v_i \, d\Omega \qquad 2.3.9$$

In matrix form:

$$[K]\{\Psi_j\} = \{F\} \qquad 2.3.10$$

The first matrix is called the "stiffness matrix" and the $\{F\}$ vector is called the "force vector".

Once the equation is presented in this way the shape and test functions have to be determined. These functions can be the same set of functions in which case a Bubnov-Galerkin scheme is said to be used. If the set of functions are different then a Petrov-Galerkin scheme is used. In this study the Bubnov-Galerkin approach was applied. So the functions v_i and f_i are the same and the equation changes to:

$$\int_{\Omega}\left(\frac{\partial f_i}{\partial x}\frac{\partial f_j}{\partial x} + \frac{\partial f_i}{\partial y}\frac{\partial f_j}{\partial y}\right) d\Omega\ \Psi_j = -\int_{\Omega}\eta\ f_i\ d\Omega \qquad 2.3.11$$

Once the domain is divided into elements the integral is evaluated for each element and then assembled. The above integral changes into:

$$\sum_{e=1}^{Ne}\int_{\Omega e}\left(\frac{\partial f_i}{\partial x}\frac{\partial f_j}{\partial x} + \frac{\partial f_i}{\partial y}\frac{\partial f_j}{\partial y}\right) d\Omega e = \sum_{e=1}^{Ne} -\int_{\Omega}\eta\ f_i\ d\Omega \qquad 2.3.12$$

where the subindex "e" means element.

The shape functions are then specified on an element basis. To do this each element is normalized into a local coordinate system (r,s) as is shown in figure 6.

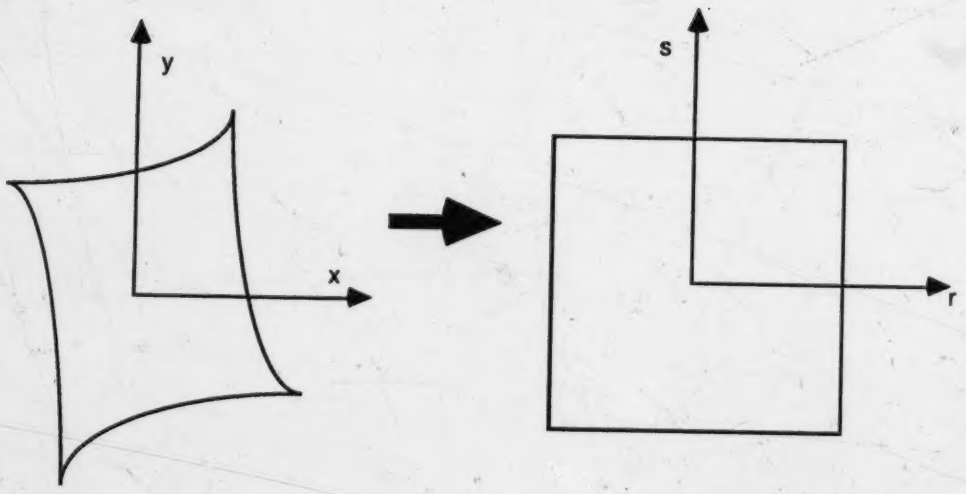


Fig. 6.- Local Coordinate System

Equation 2.3.12 shows that the values that are needed have to be a function of the global coordinate system (x,y) so a relation between the local coordinate system and the global coordinate system has to be found, this is done using the chain rule:

$$\begin{Bmatrix} \frac{\partial f_i}{\partial r} \\ \frac{\partial f_i}{\partial s} \end{Bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial y}{\partial r} \\ \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} \end{bmatrix} \begin{Bmatrix} \frac{\partial f_i}{\partial x} \\ \frac{\partial f_i}{\partial y} \end{Bmatrix} \qquad 2.3.13$$

any "x" and "y" may be expressed as:

$$x = f_j X_j$$

$$y = f_j Y_j$$

where f_j represents the set of shape functions and X_j and Y_j represent the coordinates of the vertices of the elements. The derivatives in the Jacobian matrix of equation 2.3.13 can then be calculated from:

$$\frac{\partial x}{\partial r} = \frac{\partial f_i}{\partial r} X_j \quad ; \quad \frac{\partial x}{\partial s} = \frac{\partial f_i}{\partial s} X_j$$

$$\frac{\partial y}{\partial r} = \frac{\partial f_j}{\partial r} Y_j \quad , \quad \frac{\partial y}{\partial s} = \frac{\partial f_j}{\partial s} Y_j$$

Substitute in equation 2.3.13 to get

$$\begin{Bmatrix} \frac{\partial f_i}{\partial r} \\ \frac{\partial f_i}{\partial s} \end{Bmatrix} = \begin{bmatrix} \frac{\partial f_i}{\partial r} X_j & \frac{\partial f_i}{\partial r} Y_j \\ \frac{\partial f_i}{\partial s} X_j & \frac{\partial f_i}{\partial s} Y_j \end{bmatrix} \begin{Bmatrix} \frac{\partial f_i}{\partial x} \\ \frac{\partial f_i}{\partial y} \end{Bmatrix} \qquad 2.3.14$$

where repeated subscript indicates summation. The matrix in the right hand side is called the Jacobian. Defining:

$$J_{11} = \frac{\partial f_i}{\partial r} X_j$$

$$J_{12} = \frac{\partial f_i}{\partial r} Y_j$$

$$J_{21} = \frac{\partial f_i}{\partial s} X_j$$

$$J_{22} = \frac{\partial f_i}{\partial s} Y_j$$

We can get the derivatives of the shape functions with respect to the global coordinates as:

$$\begin{Bmatrix} \frac{\partial f_i}{\partial x} \\ \frac{\partial f_i}{\partial y} \end{Bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}^{-1} \begin{Bmatrix} \frac{\partial f_i}{\partial r} \\ \frac{\partial f_i}{\partial s} \end{Bmatrix} \qquad 2.3.15$$

or:

$$\begin{Bmatrix} \frac{\partial f_i}{\partial x} \\ \frac{\partial f_i}{\partial y} \end{Bmatrix} = \frac{1}{|J|} \begin{bmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11} \end{bmatrix} \begin{Bmatrix} \frac{\partial f_i}{\partial r} \\ \frac{\partial f_i}{\partial s} \end{Bmatrix} \qquad 2.3.16$$

where $|J|$ is the value of the determinant of the jacobian. Finally, for each element:

$$d\Omega = dx\, dy = |J|\, dr\, ds$$

Once an element is chosen and the set of shape functions is defined then the entries of the Jacobian can be calculated as well as the partial derivatives of the shape functions with respect to the local coordinates (r,s). Knowing all this the integrand for each element is known. The integration that has to be computed is very complex so numerical integration is required. The number of integration points required depends on the type of element. In this study quadratic triangular elements were used with 3 integration points (Stasa, 1985).

II.3.- BOUNDARY CONDITIONS

Figure 2 shows the boundary conditions that are applied in the solution of this problem. At the free surface there is an extra condition that has to be satisfied, this condition comes from the energy equation and it refers to the value of the tangential velocity, which is equal to:

$$V=\sqrt{2g(H-y)}$$

Since what it is going to be compared are the numerical velocities and the velocities obtained from the energy equation it is very important to get good numerical velocities at the free surface.

The boundary conditions may be summarized as:

On A-B, B-C and C-D	----------	$\Psi = q$
On A-B and C-D	----------	$V=\sqrt{2g(H-y)}$
On F-E	----------	$\Psi = 0$
On A-F and D-E	----------	$\frac{\partial \psi}{\partial n} = 0$

At the free surface the value of the pressure head should be equal to zero. To estimate a correction for Y when it is not, take the partial derivative with respect to the depth Y from equation 1.1.4:

$$\frac{1}{\gamma}\frac{\partial P}{\partial Y}=\frac{\partial H}{\partial Y}-\frac{\partial}{\partial Y}\left(\frac{q^2}{2gY^2}\right)-\frac{\partial Y}{\partial Y} \qquad 2.3.1$$

or

$$\frac{\partial P}{\partial Y} = -\left(\frac{\gamma q^2}{gY^3}\right) - \gamma \qquad 2.3.2$$

which can be expressed as:

$$\Delta Y = \frac{\frac{\Delta P}{\gamma}}{\frac{q^2}{gY^3} - 1} \qquad 2.3.3$$

When the flow is solved for the first time the pressure at the free surface will not be equal to the atmospheric pressure because the values that were supposed (like the discharge and the initial guess for the free surface) are not correct. A correction from equation 2.3.3 can be made. First for the free surface control nodes the pressure head has to be found using equation 1.1.4, this can be done because the total energy, the normal velocity and the position for the nodes are known. A tolerance is set in terms of the summation of the difference in pressure at all the free surface control nodes. Actually the correction made is not exactly ΔP; it is $\alpha \Delta P$ ($\alpha \leq 1$), where α is a relaxation factor. The relaxation factor is needed because if all the ΔP is applied there is a risk that the solution will tend to jump to subcritical flow.

The flow field is solved for the discharge and the free surface profile for a given gate opening "a" and energy level "H".

To begin, a discharge "q" and a free surface profile have to be guessed. Once the solution is found for that particular discharge a correction is made.

II.4.- BOUNDARY VELOCITIES CALCULATIONS

The stream function is defined such that:

$$u = \frac{\partial \psi}{\partial y} \qquad \text{and} \qquad v = \frac{-\partial \psi}{\partial x}$$

where ψ is the stream function.

To have a velocity tangential to the free surface means that the derivative of the stream function there has to be computed. An approximation for this could be just to take the increments of the stream function at the nodes adjacent to the free surface. In fact this is a common practice in obtaining derivative calculations from finite element solutions.

As mentioned before, in this study the tangential velocities at the free surface need to be very accurate so the simple approach above would not be satisfactory unless the grid spacing were very small. The formulation used in this study is the one developed by G.F.Carey (1982). This technique yields highly accurate results as has been demonstrated by Carey, Chow and Seager (1985). It is based in the fact that the boundary derivative terms already appear in the weak statement as the natural boundary conditions. Since in this problem Dirichlet data are used on the free surface portion of the boundary, these terms were discarded. But now those are the values that need to be computed. Recalling the weak statement:

$$\int_{\Omega}\left(\frac{\partial v_i}{\partial x}\frac{\partial \Psi}{\partial x}+\frac{\partial v_i}{\partial y}\frac{\partial \Psi}{\partial y}\right)d\Omega - \int_{\Omega} v_i \eta \, d\Omega = -\int_{\Gamma} v_i\left(\frac{\partial \Psi}{\partial x}n_x + \frac{\partial \Psi}{\partial y}n_y\right)d\Gamma$$

Denote the normal derivative boundary terms (velocities) as "u". This velocity is a function of position on the boundary, so a lagrangian interpolation to this function would be:

$$u_h(s) = \sum_{m=1}^{M} U_m g_m(s)$$

where g_m is a set of basis functions and U_m are nodal values of "u". Substitute this expression into the weak statement, as well as the approximation for the stream function mentioned before to obtain:

$$\int_{\Omega}\left(\frac{\partial f_i}{\partial x}\frac{\partial f_j}{\partial x}+\frac{\partial f_i}{\partial y}\frac{\partial f_j}{\partial y}\right)d\Omega\ \Psi_j - \int_{\Omega}\eta\ f_i\, d\Omega = \sum_{m=1}^{M}\left(\int_{\Gamma} g_m(s)\, f_i(s)ds\right)\ U_m$$

furthermore, if the set of boundary basis functions "g" is chosen to be equal to the one dimensional projection of the set of basis functions "f" then on an element basis:

$$-\sum_{m=1}^{M}\left(\int_{\Gamma} f_m f_i ds\right)\ U_m = K_{ij}\,\Psi_j - F_i \qquad 2.4.1$$

with $m=1,2,....M$ M is the number of boundary basis functions,

$i=1,2,....t$ t is the number of boundary nodes

associated with U_m.

j is adummy index.

It is noted that K_{ij} j = 1,2,....,n and F_i are the preassembled entries in the finite element system for the boundary node "i", but as mentioned before this values are discarded when the boundary conditions are enforced since the value of the stream function at the boundary nodes is known. Once the solution for the stream function is obtained then the multiplication is performed, resulting in a vector containing "residuals", specified by the right hand side of equation 2.4.1. It can be expressed as:

$$TU = r$$

where U will be the boundary velocities tangential to the boundary, T is the matrix that is obtained when the left hand side of equation 2.4.1 is assembled and r is the residual vector mentioned before. Once this system of equations is solved very accurate velocities are determined.

II.5.- WALL VORTICITY COMPUTATION

In this section it will be explained how the wall vorticity is estimated.

Recalling the definition of vorticity:

$$\eta = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$$

but:

$$u = \frac{\partial \psi}{\partial y} \qquad \text{and} \qquad v = -\frac{\partial \psi}{\partial x}$$

so near a solid boundary:

$$-\eta = \frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = \frac{\partial^2 \psi}{\partial n^2}\Big|_s \qquad 2.6.1$$

where "n" is a vector normal to the boundary at point "s".

Consider figure 7, if a central finite difference approximation is used then :

$$\eta_o = \frac{\psi_1 - 2\psi_0 + \psi_{-1}}{h^2} \qquad 2.6.2$$

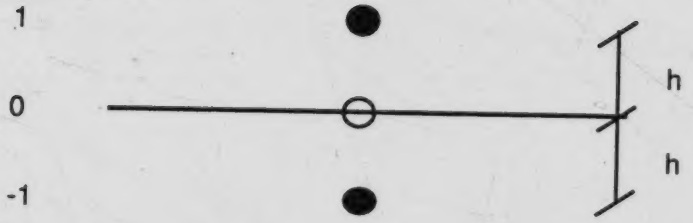


Fig.7.- Diagram to calculate the Wall Vorticity

and from a no-slip boundary condition:

$$\frac{\partial \psi}{\partial n} = \frac{\psi_1 - \psi_{-1}}{2h} = 0 \qquad \Rightarrow \qquad \psi_1 = \psi_{-1}$$

so:

$$\eta_o = \frac{-2}{h^2}(\psi_0 - \psi_1) \qquad 2.6.3$$

where η_o will be the "wall vorticity" at node "0".

Let us applied this to the sluice gate problem with the finite element approximation.

Consider figure 8 and take elements 1 and 2 for example (fig.9). Using equation 2.6.3 it can be noted that in this case, "ψ_0" will be zero at all the locations where the formula will be applied because node "0" will always be on the boundary Referring to figure 7, "h" will be just the distance between node "0" and node "1"; so taking the first node in the example (node 1 in figure 9) we have:

$$\eta_1 = \frac{-2}{y_3^2}(-\psi_3)$$

Note that node "3" is being used and not node "2", this is because this "wall vorticity" will be applied within the whole element so we think that it is more accurate to use the values for the extreme nodes at the element.

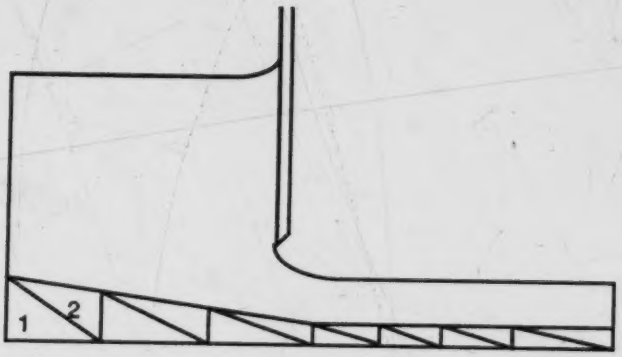


Fig.8.- Elements along the bed.

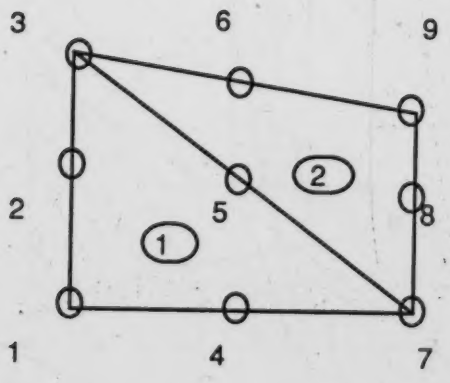


Fig. 9.- Close up from elements 1 and 2.

For node "4" it would be:

$$\eta_4 = \frac{-2}{y_6^2}(-\psi_6)$$

and for node "7"

$$\eta_7 = \frac{-2}{y_9^2}(-\psi_9)$$

What we have now is three values for the wall vorticity within elements 1 and 2, so we take the average and use this value as the vorticity that corresponds to both elements 1 and 2. Using the whole vorticity would be too much because it would imply that suddenly, at the top of the element, the value of the vorticity would go from zero to the value of the wall vorticity obtained, so a relaxation factor is used. Using half the wall vorticity would mean that a linear interpolation is being made between zero and the value obtained, the results showed that this value is still too high. Further research is needed in order to calibrate the model. The results obtained in this study are shown in chapter 4.

II.6.- DISCHARGE ITERATION

When the flow is solved for a given discharge, the velocity at the tip of the gate computed numerically, may be different from the velocity dictated by the specified energy level. The discharge has to be altered to bring the calculated energy level closer to the specified one. This is done using a Newton-Raphson method proposed by Heng et al. (1986). The method consists of the following steps:

1.- Assume a discharge q_1 and solve the flow field until convergence for the free surface is obtained, then at the tip of the gate compute:

$$f_1 = (V_{num} - V_{energy})_1$$

where:

$$V_{energy} = \sqrt{2g(H - y)}$$

V_{num} is determined using a different method that for the rest of the free surface. Once the flow field has a solution for the given discharge, a refinement in the mesh is done in the vicinity of the gate. The velocities at the integration points for the elements that contribute to the node that represents the tip of the gate are computed according to:

$$u = \frac{\partial f_i}{\partial y} \Psi_j$$

and

$$v = \frac{\partial f_i}{\partial x} \Psi_j$$

finally

$$V_{num} = \sqrt{u^2+v^2}$$

2.- Alter the discharge by a small amount δq, in this case it is being multiplied by 0.99, and obtain:

$$f_2 = (V_{num} - V_{energy})_2$$

3.- Use q_1 and q_2 to find an improved value for q according to Newton's method:

$$q_{new} = q_1 - \frac{f_1}{f_1'}$$

where:

$$f_1' = \frac{f_2 - f_1}{q_2 - q_1}$$

The number of iterations needed depends on the tolerance and on the first estimate for q. For a potential flow solution usually 5 to 7 iterations are needed. If vorticity is included, 7 to 9 iterations are typically done.

III .- THE PROGRAM

Figure 10 shows the flow chart of the program, which consists of 20 different subroutines which can be divided as follows:

1.- One input subroutine (PARDEF).
2.- Three output subroutines (ENC, OUT, OUTMAC).
3.- Two subroutines to create the mesh (SUPMAS, MESH).
4.- One global assembly subroutine (ASEMBL).
5.- Three local assembly subroutines (TRELEM,ELEM2, RECTNG).
6.- Two boundary conditions subroutines (BDRY, SURF).
7.- Two equation solvers (UACTCL,SOLVE).
8.- One subroutine to calculate the integration points (GAUSS).
9.- One subroutine to compute the new coordinates (NEWCOR).
10.- One subroutine to recalculate the relaxation factor (RELAX).
11.- One subroutine to compute the vorticity (DINVOR).
12.- Two subroutines to compute the new value for the discharge (DIS, QCHAN).

Each of these subroutines is discussed below.

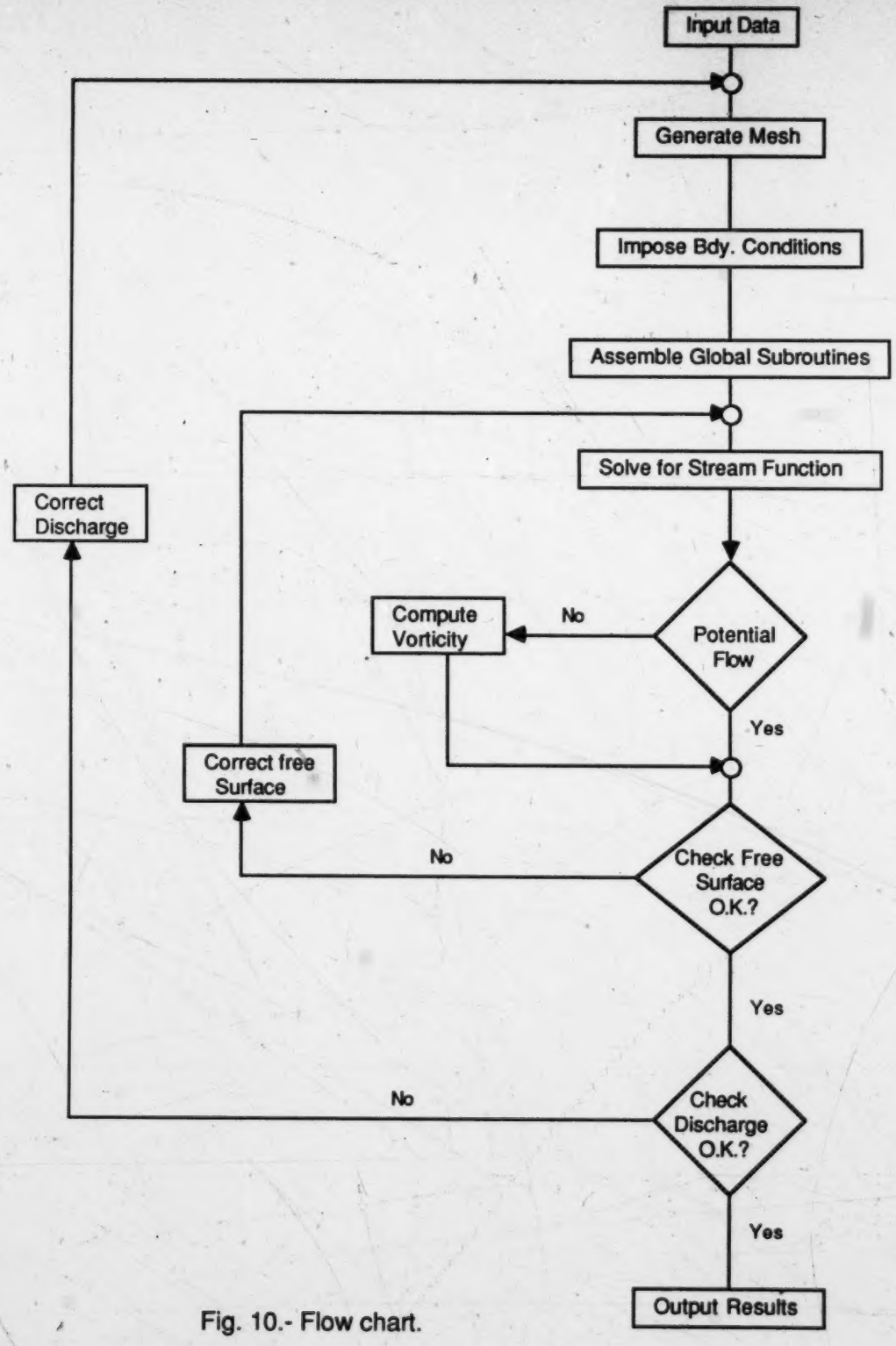


Fig. 10.- Flow chart.

III.1.- INPUT SUBROUTINE

III.1.1.- Subroutine PARDEF

In the input subroutine PARDEF all the parameters that are used in the program are read from the input data file. Basically these data can be divided in three groups:

a.- Data meant for the creation of the mesh

b.- Data corresponding to the flow itself and

c.- General control data.

III.1.1.a.- Mesh Data

The data that are used in the mesh generator are basically control parameters that indicate how many master elements are going to be used, how many elements are wanted within each master element, and the coordinates of what in this study is called the "supermaster elements". The information related to this that is read in PARDEF is:

NMASX.- Number of master elements in the "x" direction.

NMASY.- Number of master elements in the "y" direction.

PARAM.- It indicates what kind of elements are used, if it is 1 that means linear triangles, 2 for quadratic triangles and 3 for 9 nodes rectangles.

MOOTH.- Number of times that the mesh will be "smoothed".

NX(I).- Number of elements within the master element "I" in the "x" direction.

NY(I).- Number of elements within the master element "I" in the "y"direction.

SUPX(I,J).- "x" coordinates for the super master element "I".

SUPY(I,J).- "y" coordinates for the super master element "I".

PCT(I,J,K).- Percentage of Yo where we want to have the master element "J" within the super master element "I".

III.1.1.b.- Flow Data

The flow data are those that are intrinsically related to the hydraulics of the problem and they are:

HEAD.- Is the total energy level.

SSS.-Is the slope of the channel.

VORIND.- Is the relaxation factor used for the vorticity.

VAL3.- Is the first guess for the discharge.

III.1.1- Control Data

The general control data are those that control which subroutines will be called, when the program will stop, and some geometric conditions that are not needed in the mesh generator but that have to be specified in order to solve the problem.

NEBCR.- Number of master elements that are placed before the tip of the gate.

NEBG.- Number of master elements that are placed before the gate.

EXAM.- Is the parameter that indicates which kind of example is going to be solved, 1 for supercritical channels, 2 for subcritical channels, and 3 for sluice gates.

ALFA1.- Initial relaxation factor used to correct the free surface nodes.

VORFLG.- Flag that indicates if Laplace equation (VORFLG = 0) or Poisson equation (VORFLG =1).will be used.

AVGFLG.- Flag that indicates if the average procedure for the velocities will be used.

LUMFLG.- Flag that indicates if lumped values to compute the velocities are going to be used.

CIRFLG.- Flag that indicates if circulation will be computed. VORFLG has to be set to 1 in order to use CIRFLG.

ITER.- Maximum number of iterations that are permitted in

order to meet the required tolerance for each discharge. If the tolerance for a discharge is not met then the program stops.

TOL.- Is the tolerance required in terms of a summation of difference in pressure at the free surface control nodes.

TOLQ.- Tolerance for the discharge.

ITERQ.- Number of iterations permitted for the discharge.

SET.- How many integration points will be specified if 9 node rectangular elements are used.

III.2.- OUTPUT SUBROUTINES

III.2.1.- Subroutine ENC

This subroutine prints the general information about the problem. The information is :

NX(I).- Number of elements in the "x" direction within master element (I).

NY(I).- Number of elements in the "y" direction within master element (I).

ITER.- Maximum number of iterations for each discharge.

PARAM.- Kind of elements that are used.

SMOOTH.- Number of smoothing desired.

NELM.- Total number of elements.

NNODE.- Total number of nodes.

ALFA1.- Initial relaxation factor.

VORIND.- Relaxation factor for the vorticity within the elements along the bed.

VORCIN.- Relaxation factor for the vorticity within the elements along the gate.

TOL.- Initial tolerance for the iterations.

VAL3.- Assumed discharge.

AVGFLG.- Flag that indicates if average velocities are used.

LUMFLG.- Flag that indicates if lumped values for the velocities are used.

III.2.2.- Subroutine OUT

This subroutine prints the final results in the following way:

|Node Number | Y | X | Numerical Velocity | Total Head | Velocity|

It also gives information such as how many iterations were needed for that particular discharge and what was the total difference in pressure in the free surface nodes. The numerical velocity is the one that is determined through the numerical algorithm and the velocity is the one from the Energy Equation, the total head is obtained from the summation of the "y" coordinate plus the velocity head using the numerical velocity.

III.2.3.- Subroutine OUTMAC

All the streamlines and the mesh are plotted using a program developed by Dr. Peter Steffler from the Department of Civil Engineering at the University of Alberta. This program is written for the Macintosh microcomputer so the output needs to be adapted in such way that it can be used by this program. Basically the information that is written is:

a).- Total number of elements.

b).- Total number of nodes.

c).- Nodes numbers and coordinates.

d).- Elements numbers, type of element and nodes belonging to each element.

It writes this information in a nondimensional form where the "y" coordinates go from "0" to "1", this is done by dividing the real coordinates over the total head "H". The "x" coordinates are also divided by "H" in order to have the same scale when the final results are plotted.

III.3.- MESH GENERATOR SUBROUTINES

III.3.1.- Subroutine SUPMAS

In this subroutine we take the basic information read in PARDEF and then create the "master elements" that will be further processed to form the mesh that will be solved.

The basic information that is given is the coordinates of 5 nodes that give shape to a rectangular "super master element", by linear interpolation and using the percentages that tell us where we want the "master elements", all the 8 nodes per master element are found and their coordinates given. These information is passed and go into the subroutine MESH where everything is reprocessed. (figure. 11)

III.3.2.- Subroutine MESH

This subroutine is the actual mesh generator, it is capable of generating linear and quadratic triangles as well as linear and 9 nodes rectangles.

As pointed by Thompson et al. (1985) the problem of generating a curvilinear coordinate system can be formulated as a problem of generating values of the cartesian coordinates in the interior of a rectangular transformed region from specified values in the boundary. When this is done by interpolating boundary values, the method is called "algebraic generation".

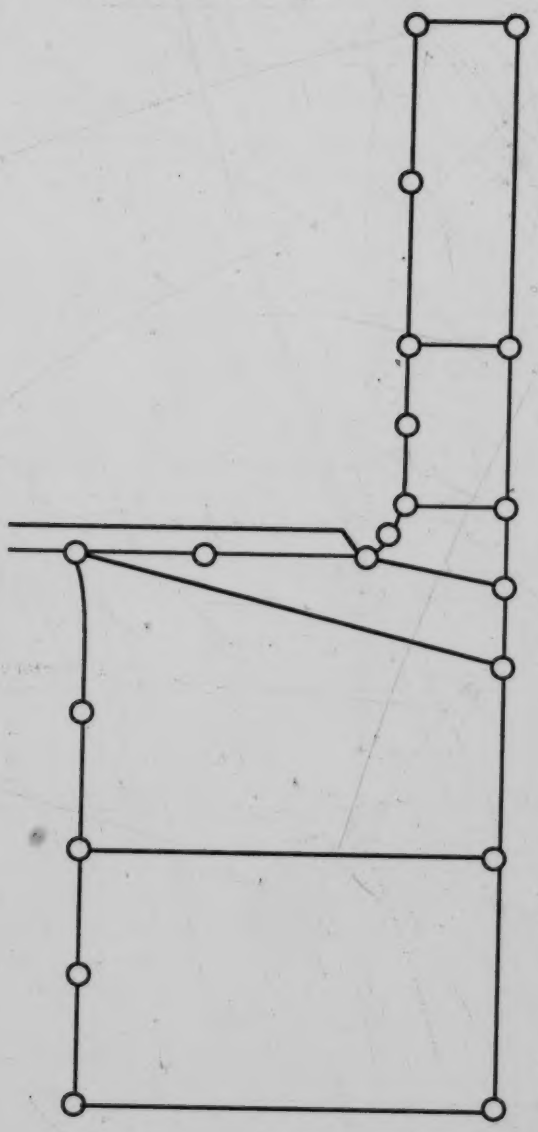


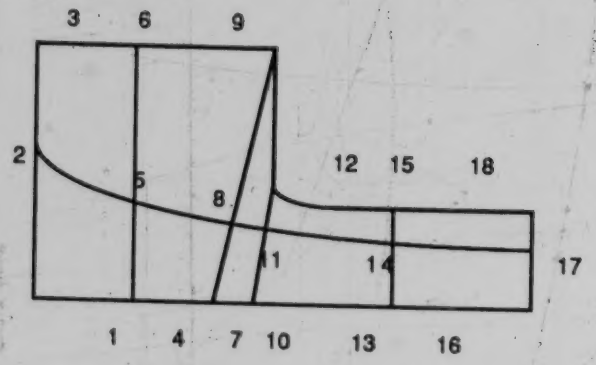
Fig. 11.- Supermaster elements coordinates.

What has to be done is to first change the physical domain into a transformed domain consisting of "patched" rectangles. In fact it could be changed into one rectangle, but by using patched master elements there is more control over the domain (fig 12)

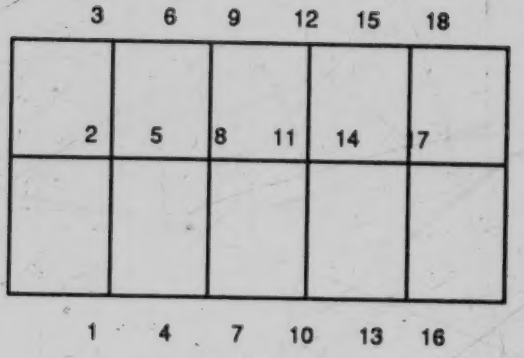
Once we have the rectangles, that are called "master elements", a local coordinate system for each master element is created. Then the interpolation for the desired number of elements that are wanted within each master element is done. This interpolation will be done by polynomials, if linear elements are wanted, a first order polynomial is used to interpolate. If quadratic, a second order polynomial will be used.

The local coordinate system will vary from (-1,-1) to (1,1) so it will be something similar to what is shown in figure 13.

For each master element some boundary values are needed. The number of boundary values depends on the type of interpolation that is used. If it's going to be first order then 4 boundary values (global coordinates) per master element are needed If we want second order interpolation then 8 boundary values per master element need to be specified. It is very important for the master elements to be compatible, that means that if they have some boundaries in common then those boundary values have to be the same.



Master Elements in the Physical Domain.



Master Elements in the Transformed Domain

Fig. 12.- Master Elements in the Physiscal and Transformed Domains.

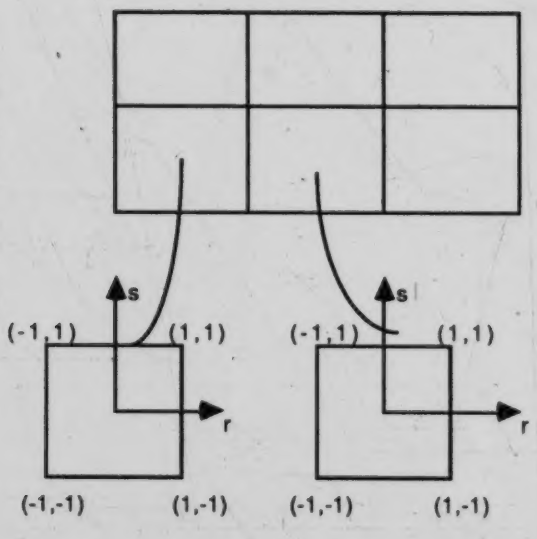


Fig. 13 .- Local Coordinates for the Master Elements

The polynomials that are used to map the local coordinates into global coordinates are kept in a "SHAPE function". These polynomials are:

First Order Interpolation.-

```
S1 = ((1-r)*(1-s)/4)*Z(L,1)
S2 = ((1+r)*(1-s)/4)*Z(L,2)
S3 = ((1+r)*(1+s)/4)*Z(L,3)
S4 = ((1-r)*(1+s)/4)*Z(L,4)
SHAPE = S1+S2+S3+S4
```

Second Order Interpolation.-

```
S1 = -((1-r)*(1-s)*(r+s+1)/4)*Z(L,1)
S2 = ((1+r)*(1-s)*(r-s-1)/4)*Z(L,2)
S3 = ((1+r)*(1+s)*(r+s-1)/4)*Z(L,3)
S4 = ((1-r)*(1+s)*(s-r-1)/4)*Z(L,4)
S5 = ((1-r^2)*(1-s)/2)*Z(L,5)
S6 = ((1+r)*(1-s^2)/2)*Z(L,6)
S7 = ((1-r^2)*(1+s)/2)*Z(L,7)
S8 = ((1-r)*(1-s^2)/2)*Z(L,8)
SHAPE = S1+S2+S3+S4+S5+S6+S7+S8
```

In both cases "Z(L,K)" refers to the "x" or "y" coordinate for the master element "L" and the node number "K". Figure 14 shows the general procedure followed to generate the mesh.

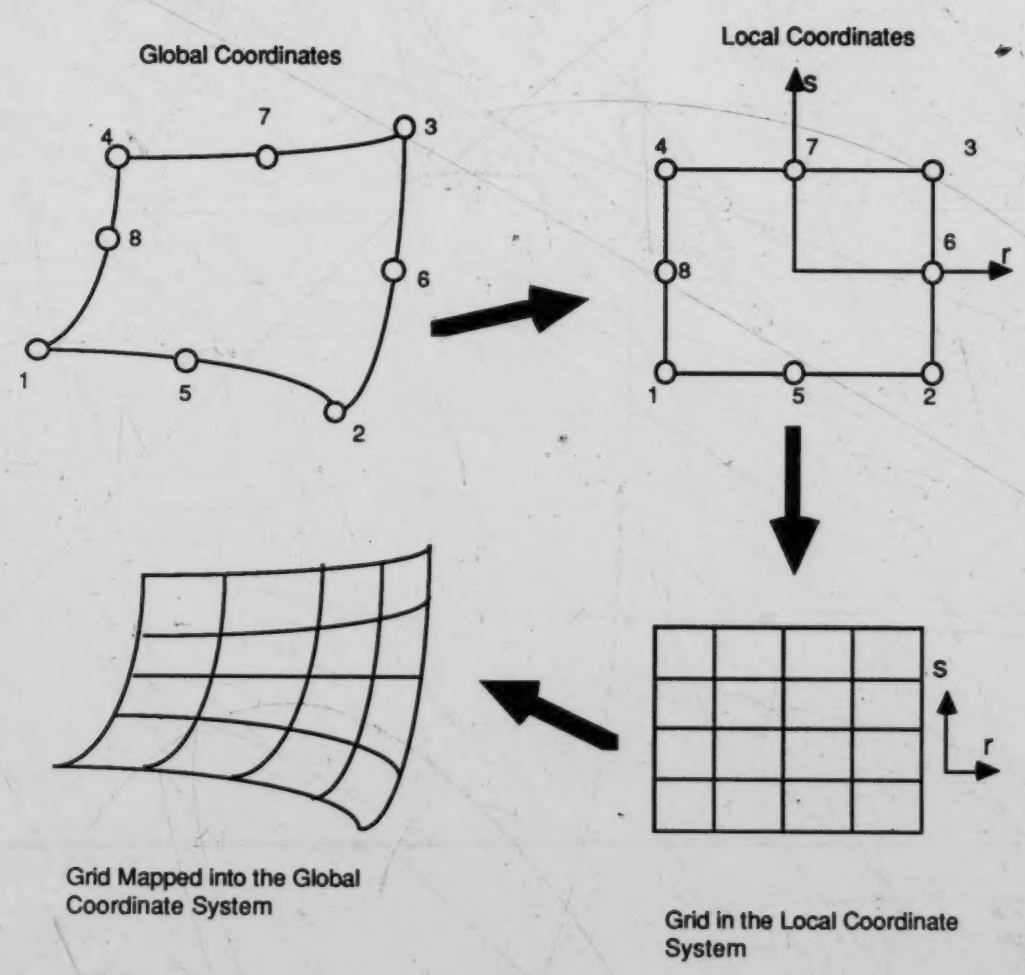


Fig. 14.- Procedure to Generate a Grid.

Now that we have all the global coordinates we may want to smooth the mesh. This is done especially if there are very abrupt changes in gradation size and in the angle of the patching elements (fig. 15)

The smoothing may be done once or twice but not more because actually what it is done when smoothing is performed, is take the average of adjacent nodes. There is a risk in smoothing, in that if there are a lot of points very close to a sharp edge, the "smoothing procedure" may cause some points to drift out of the physical boundary. This has to be checked when the plot of the mesh is inspected.

Once this is done what we actually have is all the nodes and their respective global coordinates, but we still do not have the actual elements. These elements have to be created as well as a "connectivity table". This means that the exact nodes belonging to each element have to be defined. This is done according to the parameter named PARAM which is the one that tells us which kind of elements are being used.

When this subroutine is finished, the mesh that is going to be used to solve the problem is generated. It should be emphasized that by using this mesh generator there is a lot of control about the size and the number of elements that will be used and very little input is needed. This is due to the supermaster element and master element concepts.

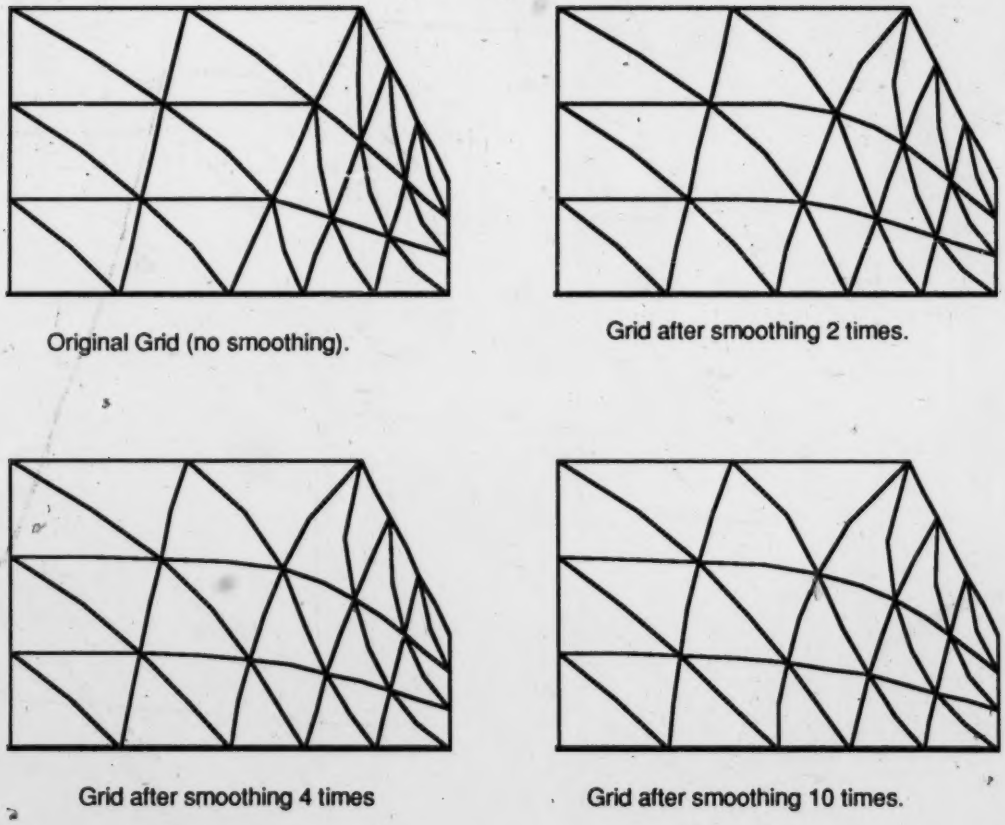


Fig. 15.- The Effect of Smoothing in a Grid.

III.4.- GLOBAL ASSEMBLY SUBROUTINE

III.4.1.- Subroutine ASEMBL

In this subroutine the global stiffness matrix as well as the nodal force vector are created.

The stiffness matrix is kept in a vector (Stasa(1985)). If this method is used another vector has to be created in order to locate the position of the diagonal of the matrix. With this method of storage, a great amount of memory can be saved. For example, if a 341 nodes problem with a half-bandwidth of 23 is solved, then 116281 locations should be needed if they are kept in a whole matrix, 7843 locations if a banded method is used (provided that the matrix is symmetric) and 5801 using skyline. In this example if half-bandwidth is used 35% more space is needed. There is no point in comparing with the storage needed if the whole matrix is kept, the advantage is obvious. Figure 16 shows a graph where the comparison between half-bandwidth and the skyline scheme are compared for the typical meshes that were used in this study. The difference may differ from mesh to mesh, the value of the half-bandwidth is a very important factor.

Once the mesh has been specified; that is, the number of nodes, the type of elements and the connectivity table are known; we can proceed to create the JDIAG vector which will locate the position of the diagonal in the skyline vector. If the stream function is being solved then the boundary subroutine (BNDRY) and the numerical integration subroutine (GAUSS) are called. After doing that for each

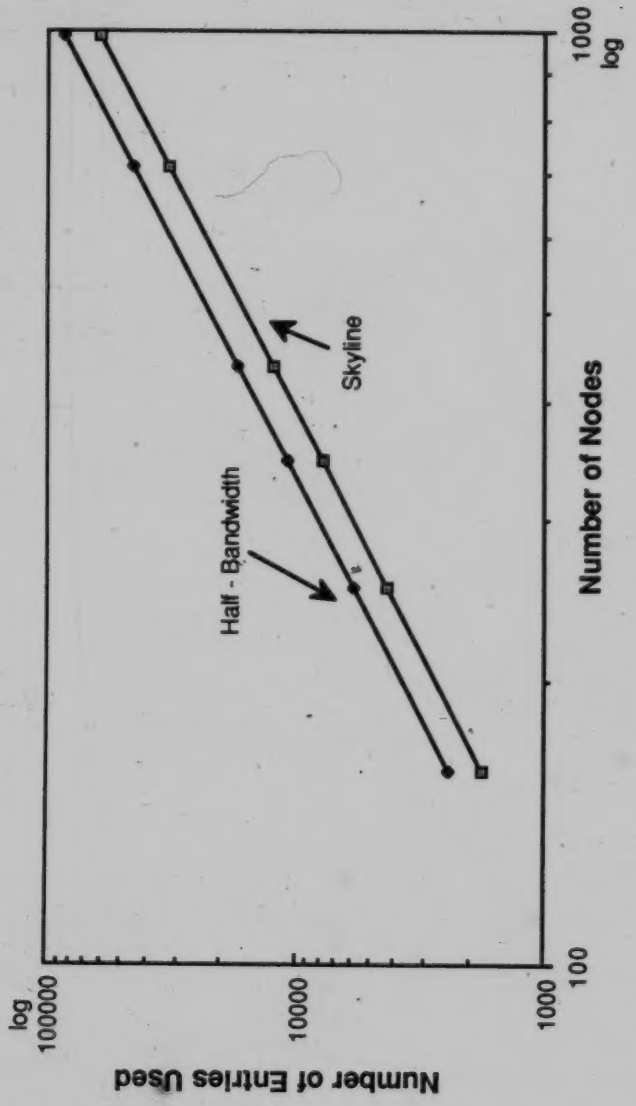


Fig. 16.- Comparison between skyline and half-bandwitdh schemes for the mesh used in the sluice gate problem.

element the respective local assembly subroutine is called. With the values obtained from the local assembly subroutines the global stiffness matrix is formed. The local assembly subroutines are called according to what elements are used and also according to what is being solved for (stream function or velocities). These local assembly subroutines will be discussed later.

When the global assembly subroutine is finished the whole system of equations is ready and one of the solvers included in the program is used to solve it.

III.5.- LOCAL ASSEMBLY SUBROUTINES

In the local assembly subroutines the Jacobian, the determinant and the partial derivatives that are needed to compute the entries for the stiffness matrix and for the force vector are calculated. The differences among this subroutines are determined by the type of element that is being used.

III.5.1.- Subroutine TRELEM

The subroutine TRELEM is the local assembly subroutine for triangular quadratic triangles. It does a numerical integration using gaussian quadrature with three integration points per element. With this number of points we can integrate with a great precision a seventh order polynomial, and the ones that are obtained from the weak statement of the finite element method for the Poisson or Laplace equations, are sixth order.

III.5.2.- Subroutine RECTNG

This subroutine is the equivalent of TRELEM but is used when 9 node rectangular elements are used. The major difference between these two subroutines is that in this one there is a way to indicate to the program how many integration points are to be used. The parameter that indicates this is called "set".

III.5.3.- Subroutine ELEM2

This subroutine is used for one dimensional quadratic elements. These elements are used in the surface velocity calculation. A five point integration rule is used to integrate the polynomials obtained from the weak statement. Fewer integration points gave inferior results.

III.6.- BOUNDARY CONDITIONS SUBROUTINES

III.6.1.- Subroutine BNDRY

In this problem, the boundary conditions numerically mean that the values for the stream function at the free surface nodes, and at the bed nodes have to be specified. This is done using the "penalty method".

Since we know that the stream function at the bottom is equal to zero, and at the free surface is equal to the assumed discharge, we give this values to the corresponding nodes in the stiffness matrix and add a very large number. At the same time we give the same values in the force vector (right hand side of the equations) and multiply them by the same large number, so when the system of equations is solved the large numbers cancel each other and the actual boundary values will remain.

If this program should be used to solve different types of flow, then different boundary conditions subroutines would have to be written.

III.6.2.- Subroutine SURF

Once the solution for the stream function is found then the tangential velocity at the free surface nodes has to be found. To do this the algorithm developed by Carey (1982) is used (see chapter II). In this subroutine the multiplication of the stiffness matrix by the stream function is performed but just for the free surface nodes. The result is kept in a "residual vector" which will be used to solve the one dimensional problem of the velocities.

At the tip of the gate the normal velocity is given by the energy level as:

$$V = \sqrt{2g(H-a)}$$

and at the stagnation point $V = 0$

These values will be given as the boundary conditions and the problem will be solved for the rest of the free surface nodes. The one dimensional elements are also created in this subroutine, these elements will be linear or quadratic according to the two dimensional elements that are used.

III.7.- SOLVER SUBROUTINES

III.7.1.- Subroutine UACTCL

This solver was first proposed by Zienkiewicz (1977). It solves a system of equations where the matrix is kept in a skyline vector. This particular version solves asymmetric systems, so two skyline vectors are used; vector A keeps the diagonal and the upper elements of the matrix and vector C keeps the lower elements of the matrix and "1" on the diagonal. It uses a back substitution scheme.

III.7.2.- Subroutine SOLVE

This solver is used if a lumped scheme is going to be applied to solve for the velocities. What is done here is the "lumping" itself, then the "lumped" vector derived is just divided by the "residual vector" that was created in the SURF subroutine and thus the velocities are obtained. Since the diagonal is heavy, there is not a considerable difference in using a lumped scheme or the normal one but there is less computational effort involved when the lumped scheme is used. Therefore this was the method applied throughout this study.

III.8.- INTEGRATION POINTS SUBROUTINE

III.8.1.- Subroutine GAUSS

When the local stiffness matrices are created for each element a numerical integration has to be performed. Since all the elements in local coordinates are equal then all the integration points will be the same for all the elements. It is more efficient to create a subroutine that will be called just once before a particular problem, that is for one discharge and for one iteration, is solved that inserting all the possible gaussian integration points in each local assembly subroutine. This is done in the subroutine GAUSS. Here, according to the elements that are used, the local coordinates for the integration points are assigned and passed to the respective local assembly subroutine. The call for this subroutine is made within the global assembly subroutine.

III.9.- NEW COORDINATES SUBROUTINE

III.9.1.- Subroutine NEWCOR

This subroutine calculates the new coordinates for the free surface control nodes. To do this first a pointer array that keeps track of the control nodes has to be created. Then, if average velocities will be used the procedure to do this is applied. Average velocities are used because the convergence to the solution is not stable and by taking a weighted average of the velocities a better convergence is obtained.

Once this is done the correction for the position is made. Only the "y" coordinate is corrected. The correction is made according to the procedure described in chapter II. First, the velocity head for the node that is being considered is computed and the total head for that node obtained (velocity head plus elevation) using the numerical velocity, then the difference in pressure is computed (original head minus numerical head) and the correction calculated. This is done for all the free surface control nodes, after doing this an update for the master elements is performed by substituting the values originally kept by the values just obtained. An error indicator is derived by adding the absolute value of the differences in pressure for all the free surface control nodes.

All the results are passed to the main program where they will be directed to the next subroutine according to the value of the error.

III.10.- RELAXATION FACTOR SUBROUTINE

III.10.1.- Subroutine RELAX

The correction computed in the subroutine NEWCOR is relaxed to have a better convergence. If the whole correction is applied there is a risk that the solution will jump. This "relaxation" is applied through a "relaxation factor" which is not constant, but varies according to the convergence that is being obtained. The difference in pressure of consecutive solutions is analyzed. If this difference is decreasing, the solution is converging and the relaxation factor is increased. Otherwise the relaxation factor is decreased. There are an upper and a lower limit for the factor. Experience showed that 15% is a convenient upper limit and correcting by less than 3% has no meaning because the consecutive solutions would be almost identical. When the lower limit is reached then the factor is upgraded to 7.5%. Experience also showed that in most of the cases with this jump in the solution, the rate of convergence suddenly increases. If the upper limit is reached then it stays steady at that value. Usually the initial relaxation factor used was 10%.

III.11.- VORTICITY SUBROUTINE

III.11.1.- Subroutine DINVOR

DINVOR is a "dynamic vorticity" subroutine. This means that after each iteration the wall vorticity for each element along the bed is recalculated. It is dynamic because the value for the vorticity will change with each iteration and at the end it will also converge. The wall vorticity is also relaxed but at a constant rate all the time. Further investigation is required in order to provide a valid relaxation factor. The value found was between 33% and 40% on the a/H ratio.

There is a flag that indicates if the wall vorticity is to be calculated in the elements along the gate as well as in the elements along the bed or just in the last ones. The VORT vector that keeps the values for the wall vorticity for each element is passed to the ASEMBL subroutine where it is used in the assembly of the stiffness matrix and force vector.

III.12.- DISCHARGE ITERATION SUBROUTINE

III.12.1.- Subroutine DIS

In this subroutine, a numerical calculation for the velocity at the tip of the gate is performed. This is done using the derivatives of the stream function at the integration points. Three points are used, the ones closer to the tip of the gate. Once the velocity at these points is obtained, an extrapolation is done. If the refinement of the mesh is appropriate, good estimations for the velocity are determined.

III.12.2.- Subroutine QCHAN

After a solution is obtained for a particular discharge, then an iteration is performed in order to obtain the right discharge. This is done following a Newton-Rapshon scheme proposed by Heng et al. (1986) as mentioned in the last chapter. With this algorithm and having a tolerance of 0.0001 cfs/ft, usually 5 to 13 iterations were needed.

IV.- ANALYSIS OF RESULTS

IV.1.- ANALYSIS OF EXPERIMENTAL RESULTS.

It has been a common practice to define the discharge coefficient as:

$$C_d = \frac{q}{a\sqrt{2gy_o}}$$

and the contraction coefficient as:

$$C_c = \frac{y_d}{a}$$

where "a" is the gate opening, y_o the far upstream depth, y_d the far downstream depth, "q" the discharge per unit width and "g" the acceleration due to gravity.

If the experimental results are analyzed it can be noted that for different gate openings but equal gate openings-head ratios, different values for the coefficients may be obtained, especially for the contraction coefficient. Figure 17 shows the discharge coefficients from the experimental results obtained by Rajaratnam (1977). If the energy equation is applied between a section far upstream of the gate and a section far downstream it can be shown that:

$$H = aC_c + \frac{q^2}{2ga^2}\frac{1}{C_c^2} \quad \text{or}$$

$$\frac{H}{a} = C_c + \frac{q^2}{2ga^3}\frac{1}{C_c^2}$$

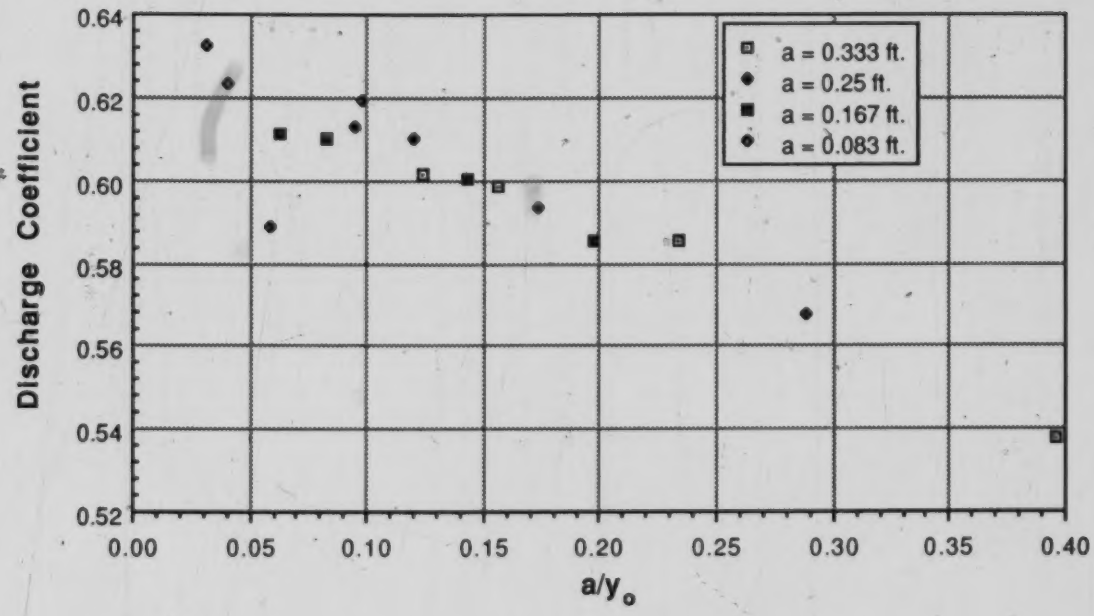


Fig. 17.- Discharge Coefficients from Rajaratnam's experimental results (1977).

We can define a "discharge parameter" as

$$C_{d2} = \sqrt{\frac{q^2}{2ga^3}}$$

For the same experimental results, a/H is plotted versus this parameter on figure 18. It seems that there is a better correlation using this relation and it has the advantage over the discharge coefficient that the far upstream depth does not need to be known.

Figure 19 shows the contraction coefficients for the same experimental results and again it may be observed that there is not a unique correlation between a/H and C_c. If the results from Benjamin (1956) and Rajaratnam (1977) for 2 different gate openings are plotted then figure 20 is obtained. It is noted that even for similar gate openings the results differ. In Rajaratnam's experiments there were made 4 sets of experiments, each for a different gate opening. The gate openings that were studied were 4, 3, 2 and 1 inches. All the details can be found in the original papers.

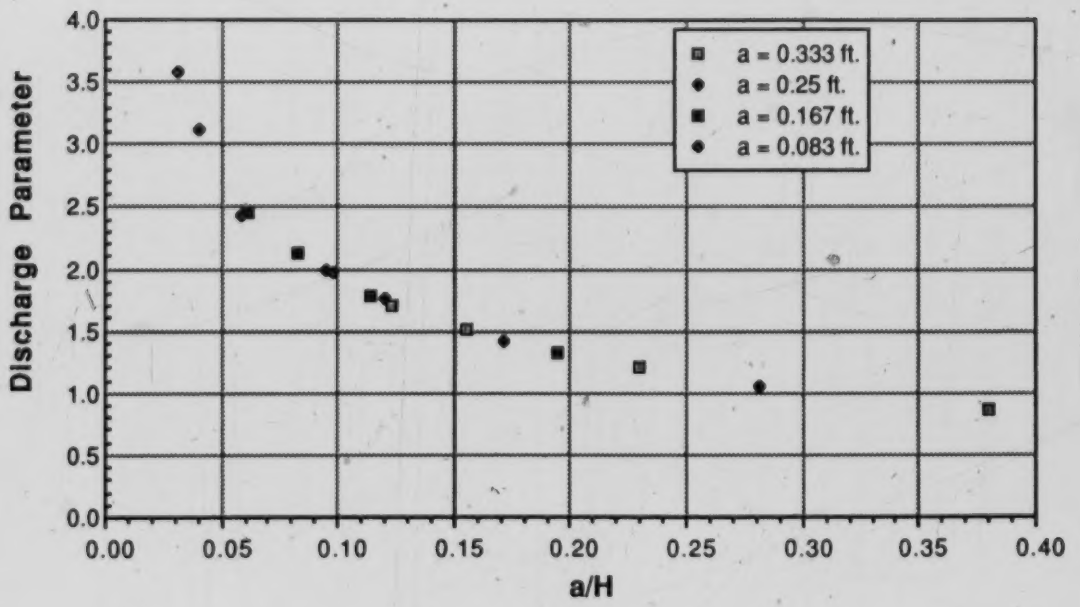


Fig. 18.- Discharge parameter from Rajaratnam's experimental results (1977).

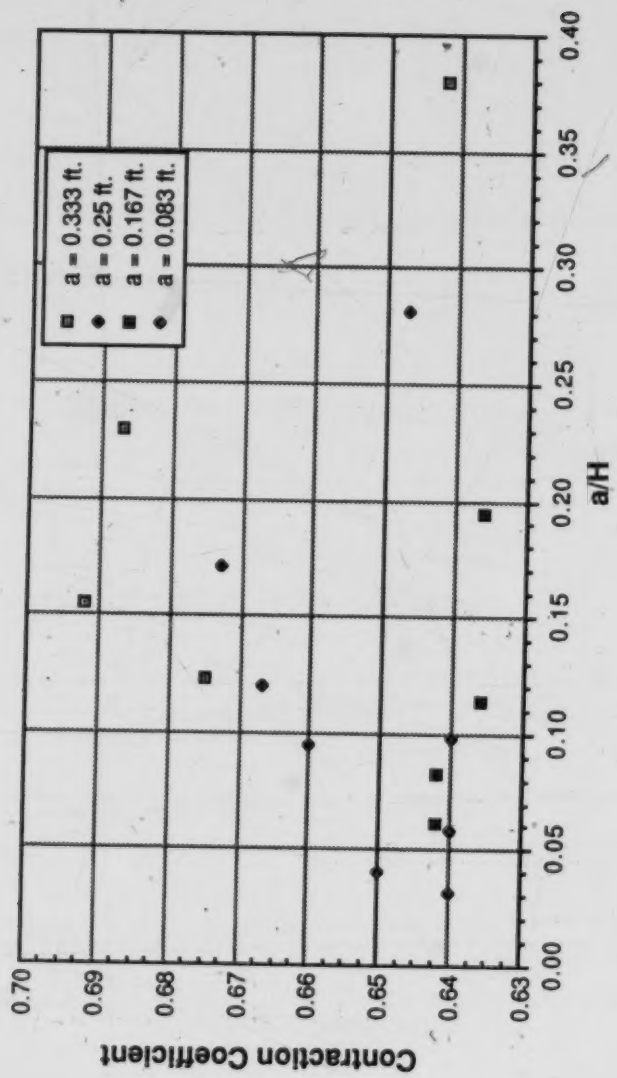


Fig. 19.- Contraction coefficients from Rajaratnam's experimental results (1977).

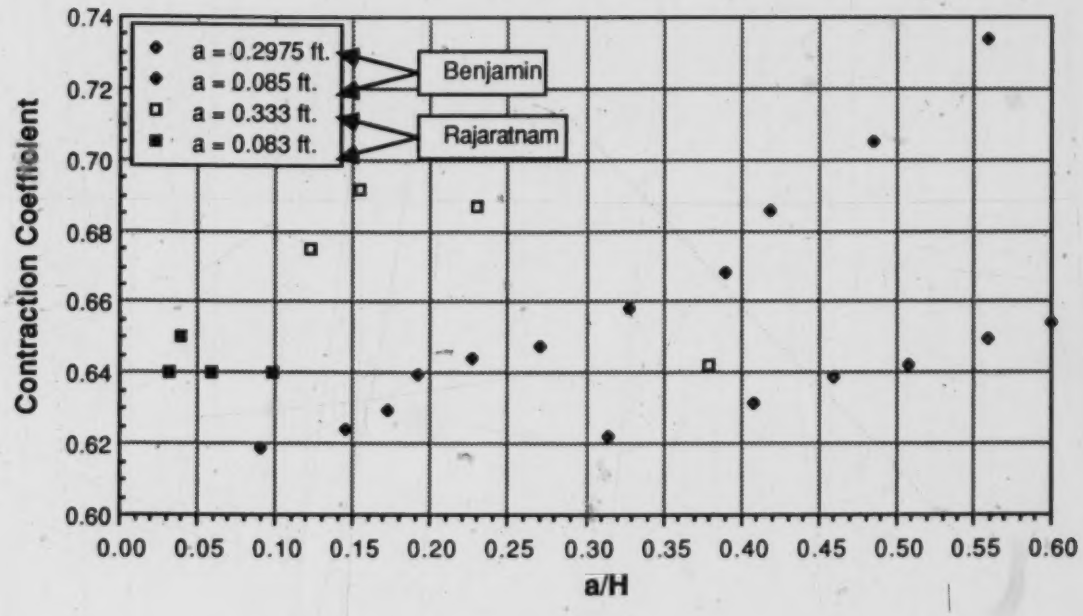


Fig. 20.- Contraction coefficient comparison between Benjamin's (1956) and Rajaratnam's (1977) experimental results.

IV.2.- POTENTIAL FLOW RESULTS.

In order to verify the method proposed in this study, it was decided to run some of the examples that had been studied before. Since all the studies done until now use models based on potential flow theory, it implied that this model had to be used with zero vorticity.

The first run was done based on Isaacs (1977) example. This one was chosen because he reported the actual values for the free surface that he obtained, so scaling from graphs was avoided. To guess the first discharge, the standard plots that appear in the literature (Chow (1959), Henderson (1966)) are used. The gate opening-head ratio was taken as 0.3 and the total energy head was 1.0 ft. The tolerance imposed in this example was such that the summation at all the free surface nodes of the difference between the prescribed and numerical energy levels was lower than 0.015 ft. The maximum difference at any node was 0.26% of the prescribed energy level. Both results are shown in figure 21 and they are in very good agreement. It is interesting the number of degrees of freedom used; in this study 115 were used. In Isaacs' study the number of degrees of freedom used were not reported, but from the figures published it seems that about 284 were used, that is 147% more degrees of freedom. By using fewer elements sometimes more iterations are needed but it has the advantage that it may be implemented in a smaller computer, since the memory requirements are less.

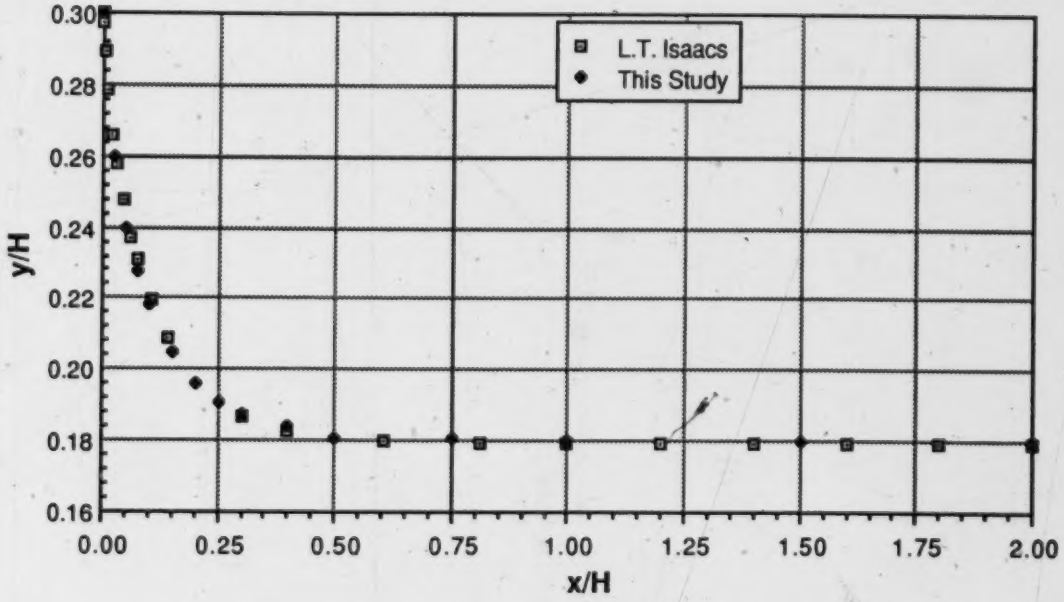


Fig. 21.- Downstream free surface profile comparison between Isaacs' (1977) and this study.

After obtaining these results, different gate openings-head ratios were tried and then compared with the results obtained by Fangmeier and Strelkoff (1968). The results for the downstream free surface profiles are shown in figure 22. The ones for the upstream free surface profiles are shown in figure 23. The results from Fangmeier and Strelkoff were scaled from the figures reported. Figure 23 shows that in this study there is some problem with the upstream free surface for a gate opening-head ratio of 0.4, but that may be because few elements were used in that portion of the flow field, and in order to reach the stagnation point a steep slope is needed.

For some reason usually the discharge coefficient is plotted versus a/y_o instead of a/H. Figure 24 shows different results for the discharge coefficient (C_d), including the experimental results from Henry (1950) and Rajaratnam (1977), all the analytical and numerical studies were done using potential flow. Figure 25 shows the discharge parameter for different studies. Figure 26 shows the contraction coefficients versus a/H for the analytical and numerical results. In figure 27 the experimental results are also plotted.

When the flow field is solved the value of the stream function at the nodes is obtained. Figure 28 shows the meshes used to solve the potential flow examples for different gate opening-head ratios. The streamlines obtained are shown in figure 29.

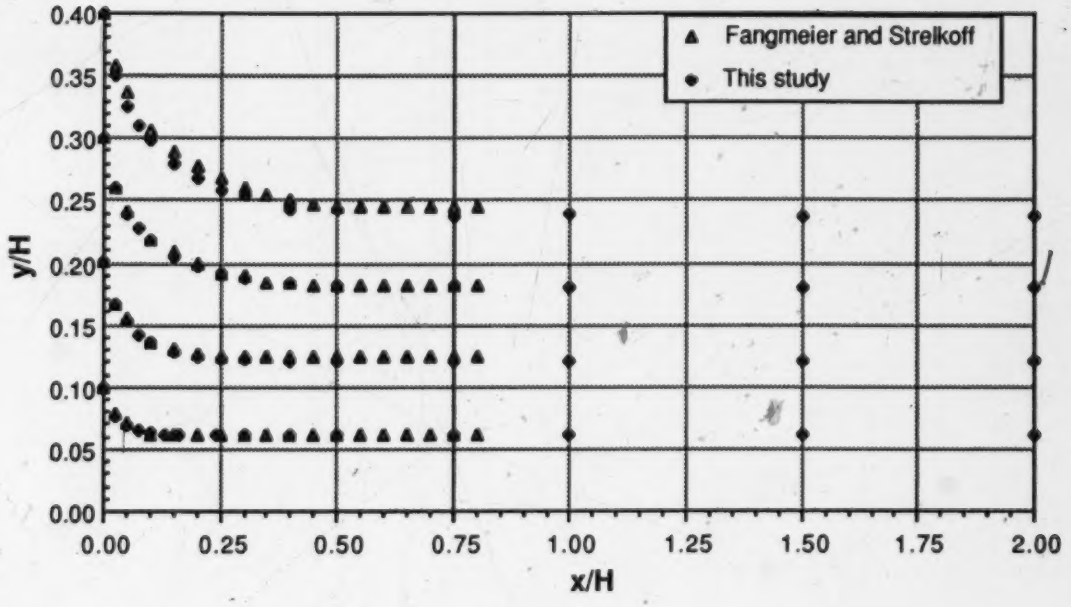


Fig. 22.- Downstream free surface profiles from Fangmeier and Strelkoff (1968) and from this study.

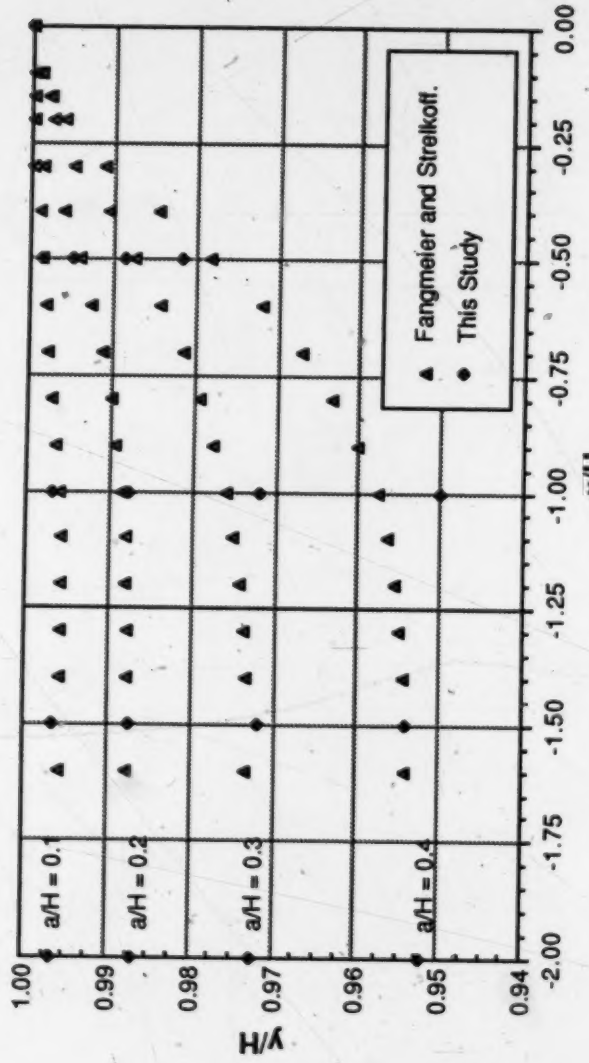


Fig. 23.- Upstream free surface profiles from Fangmeier and Strelkoff (1968) and from this study.

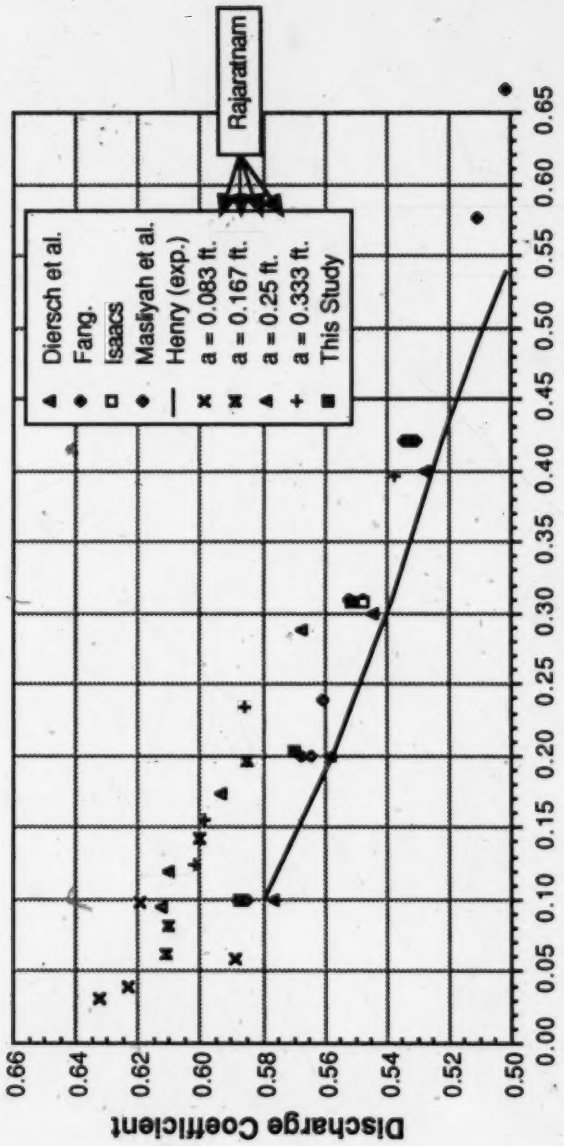


Fig. 24.- Discharge coefficients from different analytical, numerical and experimental studies.

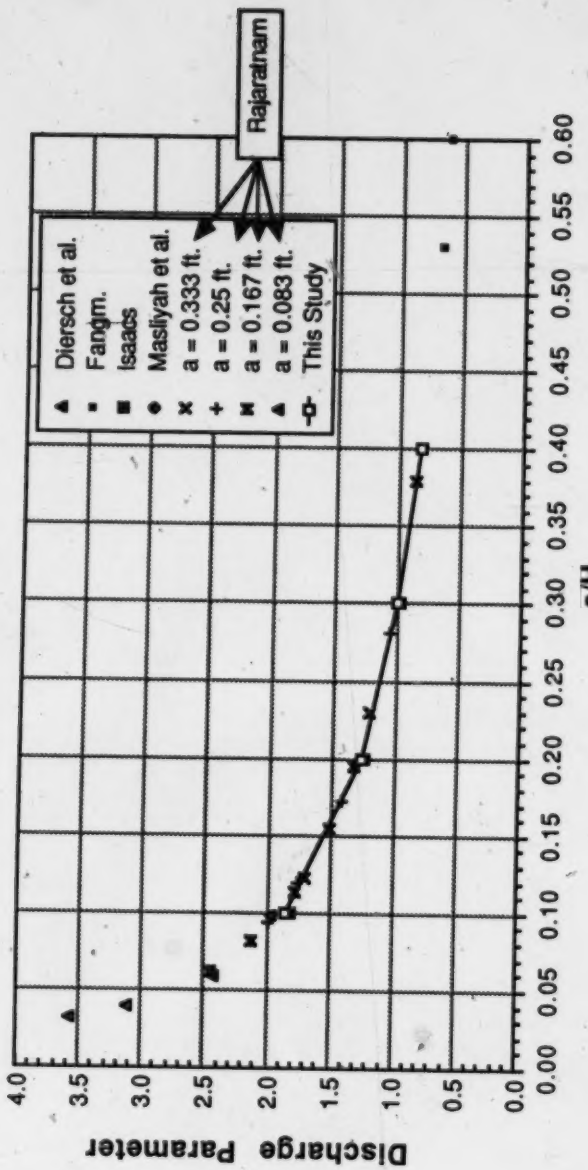


Fig. 25.- Discharge parameter from different analytical, numerical and experimental studies.

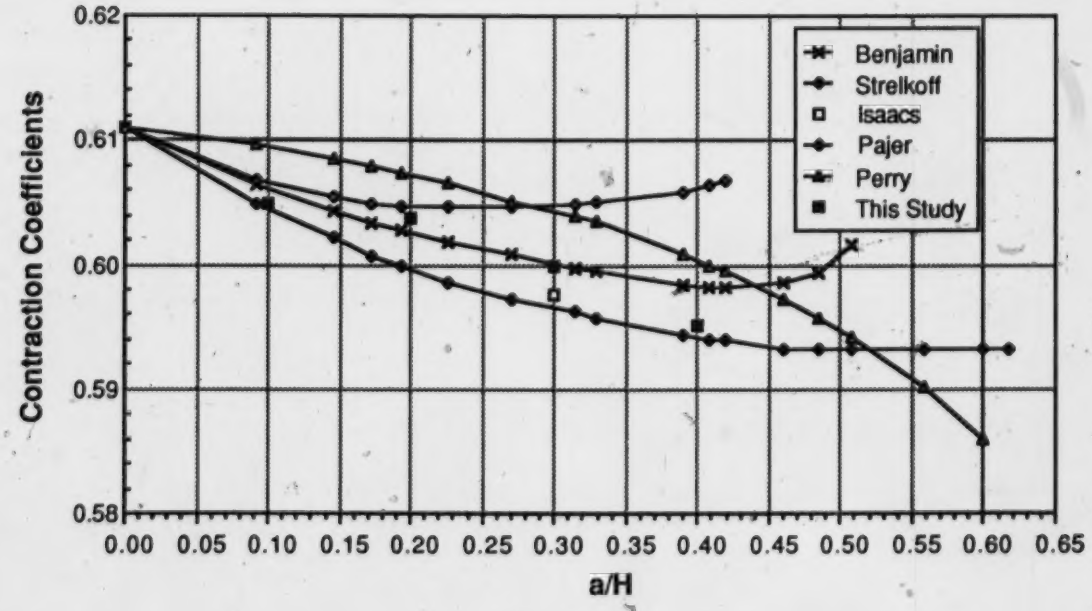


Fig. 26.- Contraction coefficients from different analytical and numerical studies.

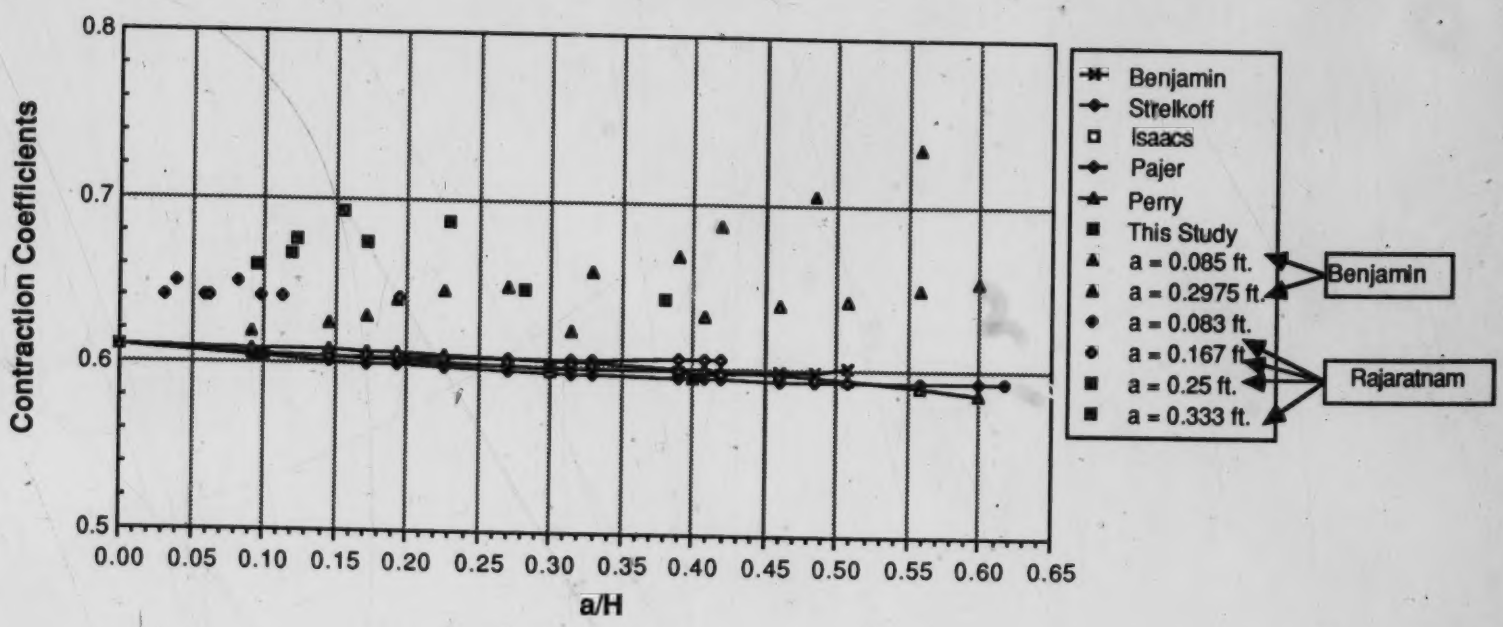
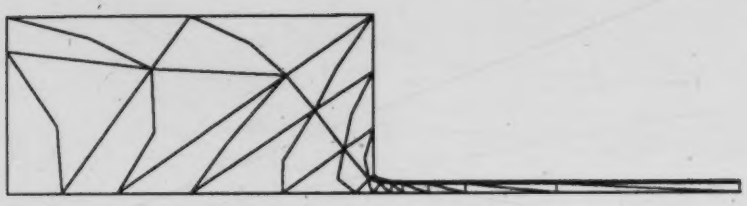
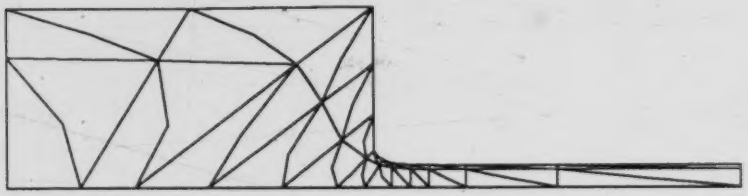


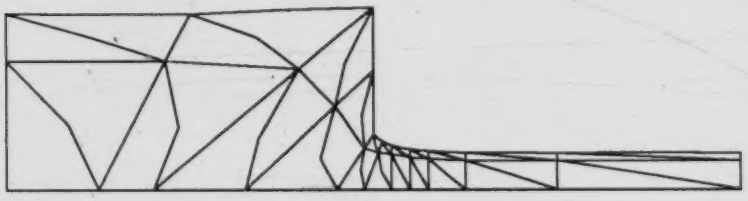
Fig. 27.- Contraction coefficients from analytical, numerical and experimental studies.



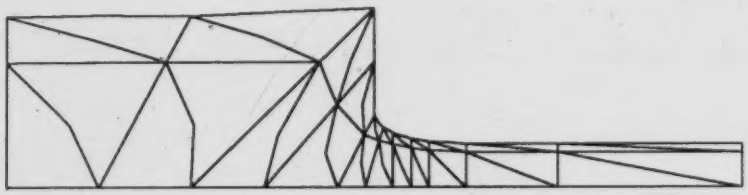
a/H = 0.1



a/H = 0.2



a/H = 0.3



a/H = 0.4

Fig. 28.- Meshes used to solve different gate opening-head ratios for potential flow problems.

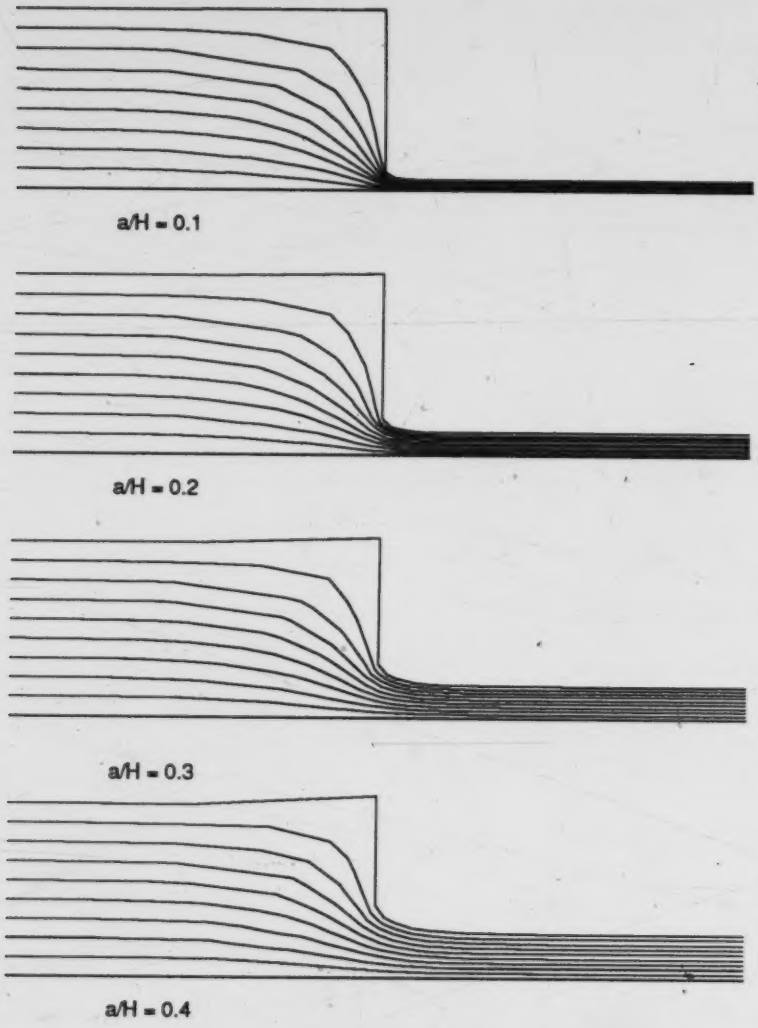


Fig. 29.- Streamlines for different gate opening-head ratios for potential flow.

As mentioned before, an advantage of this method is that the number of elements that are needed to get good results is considerably less than what has been reported in other studies; the same example (for a/H = 0.3) was run using more elements and the results are:

# of elements	#of nodes	C_d	C_{d2}	C_c	CPU
44	115	0.5513	0.9925	0.5999	5s.
176	405	0.5510	0.9920	0.5996	65s.
396	871	0.5509	0.9917	0.5998	300s.

as we can see there is no considerable difference in the results, and the computational time that is saved is enormous.

IV.3 RESULTS WHEN VORTICITY IS INCLUDED.

As it was mentioned in earlier chapters a new feature for this study is the introduction of vorticity into the equations. When vorticity is used there is a tendency for the downstream flow to rise, thus the contraction coefficients that are obtained are higher than the ones obtained when potential flow is considered. For the discharge it seems that there is not a considerable difference, especially if the discharge parameter (C_{d2}) is used instead of the discharge coefficient (C_d).

Two different experiments were done. In the first set, the discharge was prescribed according to the results from the potential flow solution. It was found that if the percentage of vorticity is incremented, the value for the contraction coefficient increases. This is shown in figure 30. We suppose that the reason for this behavior is the rotationality of the flow. As the flow develops downstream, the vorticity is increasing until it reaches a certain value where the flow is uniform again. Figure 31 shows for the case of a gate opening-head ratio of 0.3 the values that were obtained with a percentage of vorticity of 0.25.

In the second set of experiments, different configurations of gate opening-head ratios were modelled using different percentages of vorticity and the discharge was also iterated.

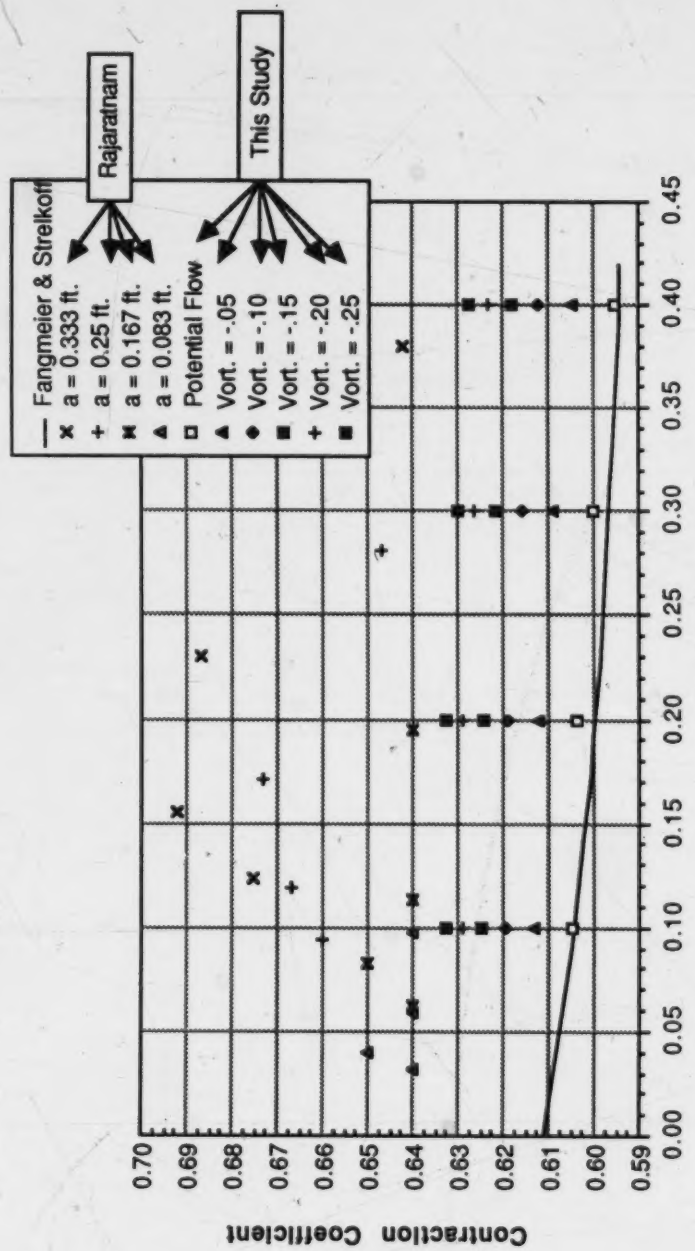


Fig. 30.- Contraction coefficients from Rajaratnam (1977), Fangmeier and Strelkoff (1968), and from this study, (for a prescribed discharge).

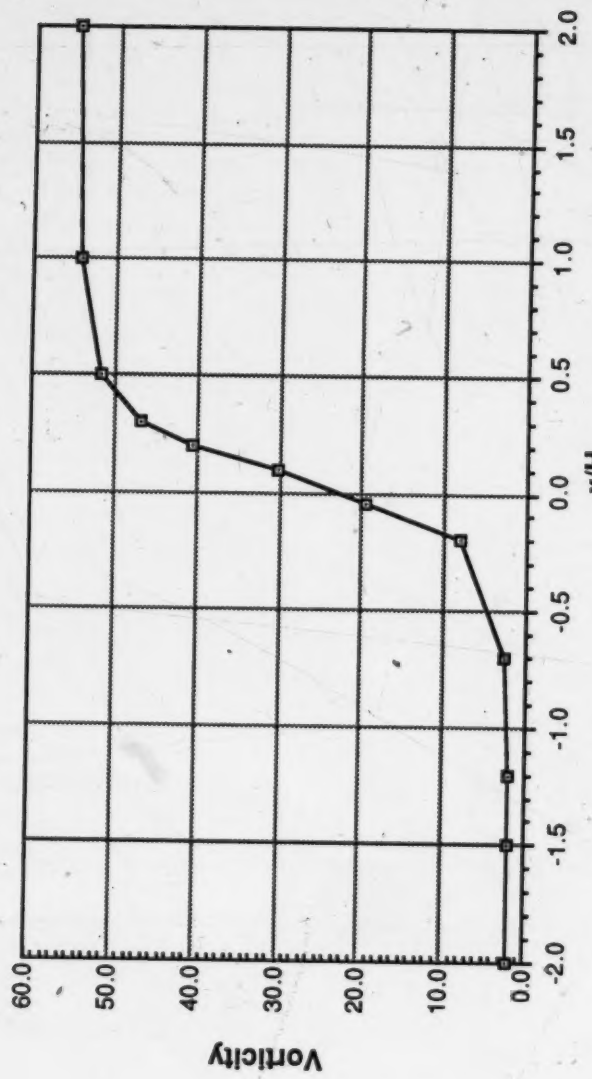


Fig. 31.- Development of vorticity within the elements along the bed.

It was noted that there is a sudden change in the coefficients from the potential flow solution to the ones with vorticity, but the changes when different percentages of vorticity are appplied are not large compared with the one that happens in the beginning. It was found that there is an upper limit of the value that may be used for the vorticity for a prescribed discharge. This value is between 0.33 and 0.4. Further research is needed in order to calibrate the model. The reason that a relaxation factor has to be used is because the velocity profile assumed is not correct, so the value of the vorticity has to be reduced. Figures 32 and 33 show the variation of the discharge parameter and the discharge coefficient for different gate opening-head ratios as the vorticity is increased. Figure 34 shows the variation of the contraction coefficient.

In figure 35 the experimental results for the discharge coefficient obtained by Rajaratnam (1977) and by Henry (1950), are plotted along with the results obtained in this study for different percentages of vorticity. The thickness of the bottom layer was 0.10 of the depth. For the same set of data, figure 36 shows the results for the discharge parameter, and figure 37 the results for the contraction coefficient.

As mentioned in chapter II vorticity is introduced dynamically as the solution is being derived, but just in the elements along the bed. It is necessary to modify the mesh that was to solve potential flow because a thin layer of elements is needed along the bed. Figure 38 shows the different meshes for different gate opening-head ratios that were used when vorticity was applied. The streamlines obtained from the solution of these meshes are shown in figure 39.

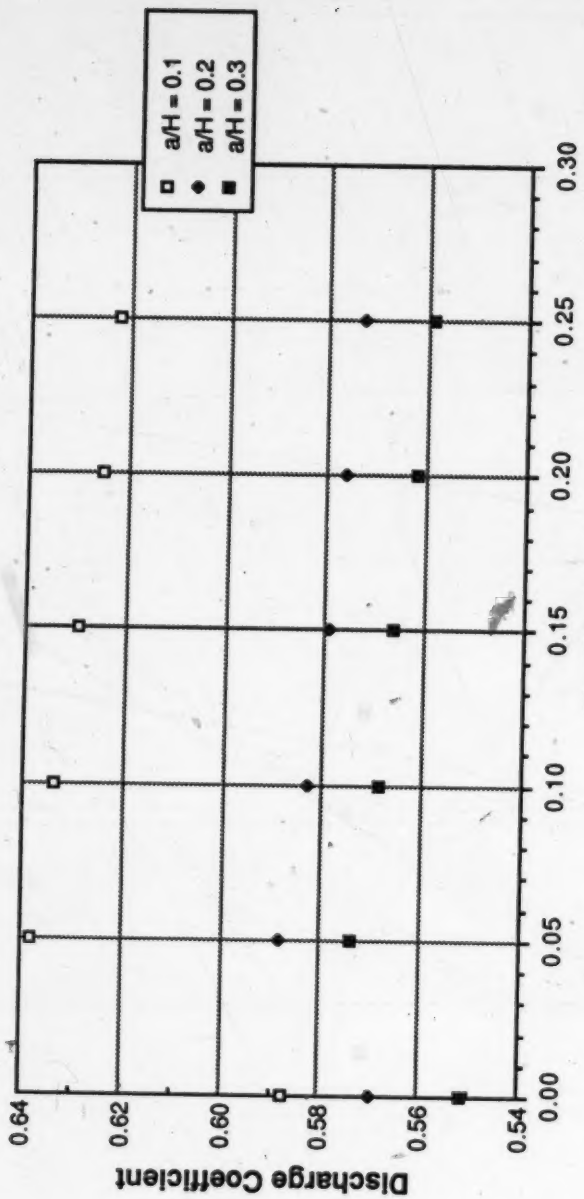


Fig. 32.- Effect of vorticity on the discharge coefficient.
Thickness of the bottom layer = 0.10 Y

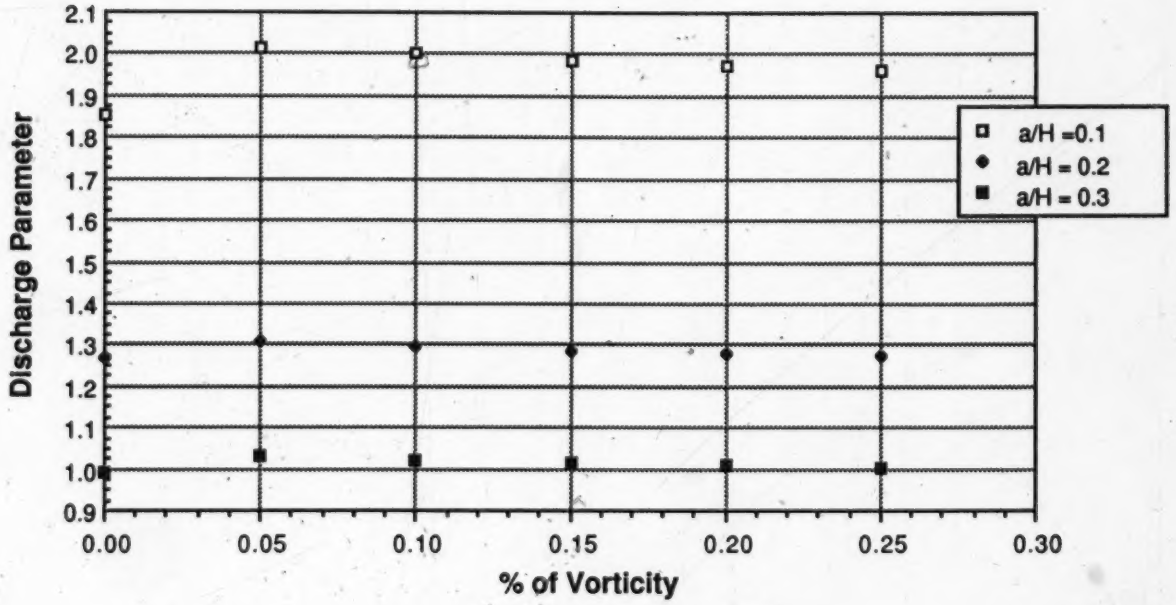


Fig. 33.- Effect of vorticity on the discharge parameter.
Thickness of the bottom layer = 0.10 Y.

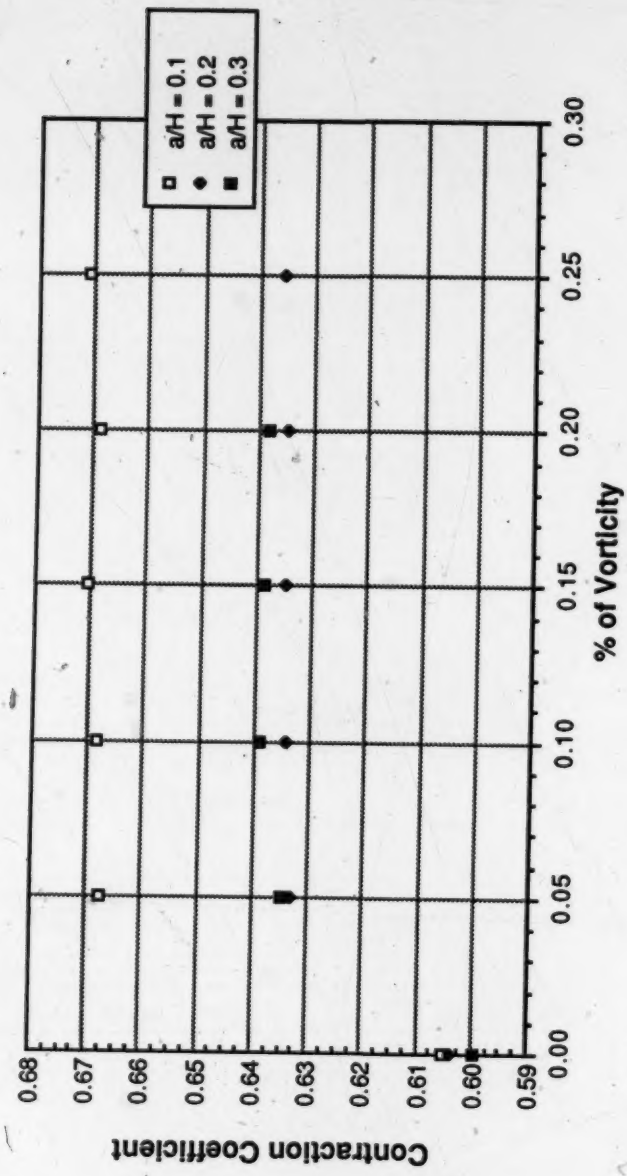


Fig. 34.- Effect of vorticity on the contraction coefficient.
Thickness of the bottom layer = 0.10 Y.

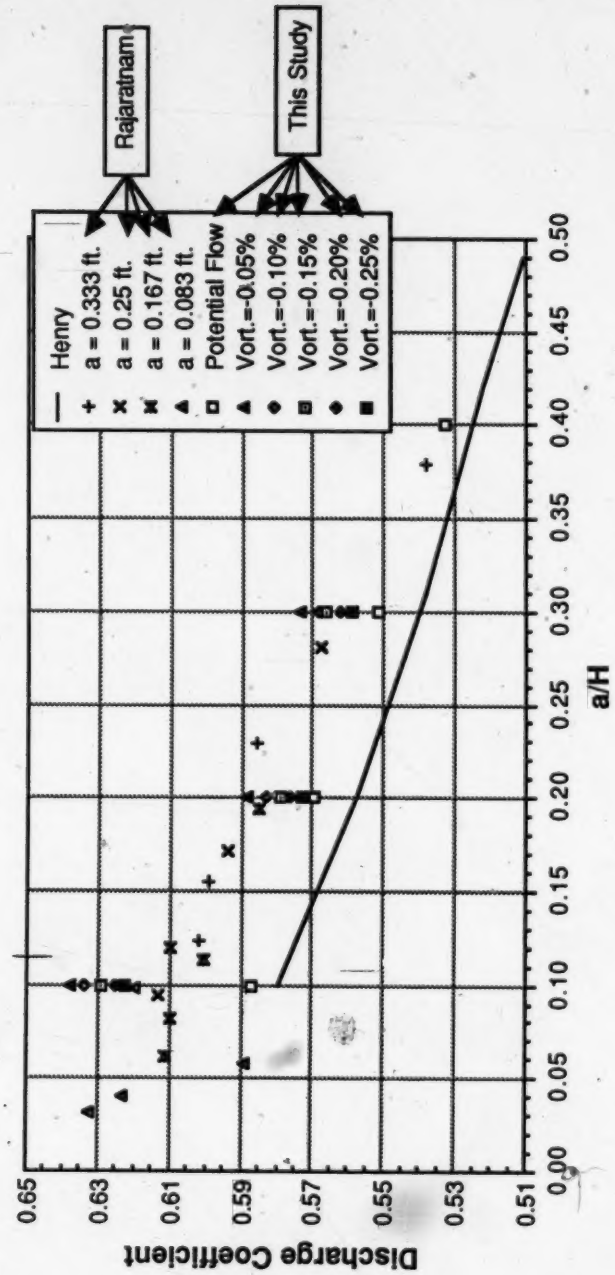


Fig. 35.- Discharge coefficients from Rajaratnam (1977), Henry (1950) and from this study. Thickness of the bottom layer = 0.10 Y.

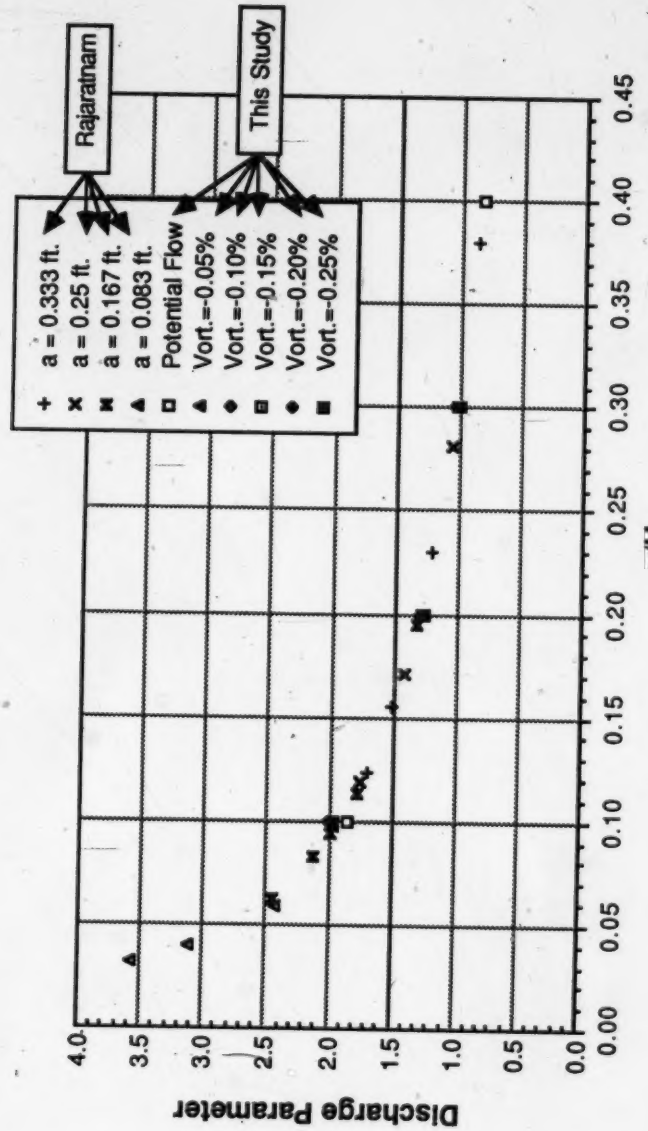


Fig. 36.- Discharge parameter from Rajaratnam (1977) and from this study. Thickness of the bottom layer = 0.10 Y.

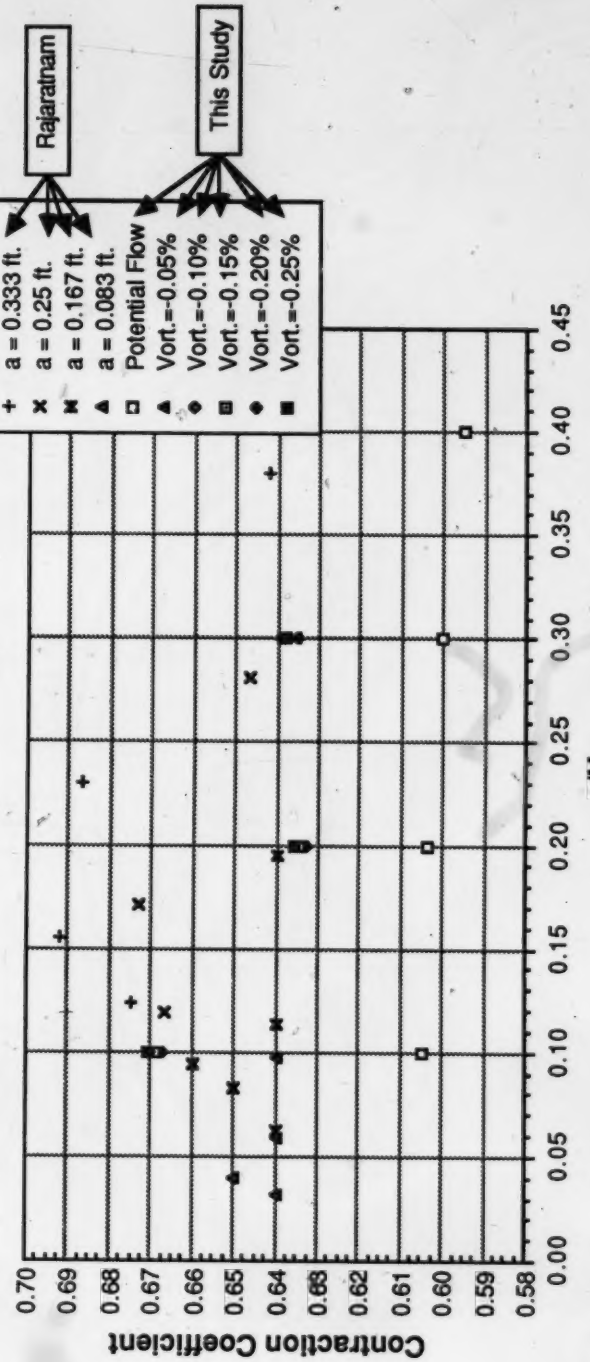
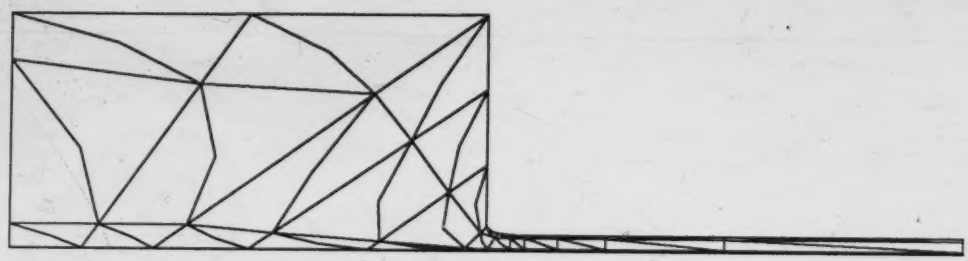
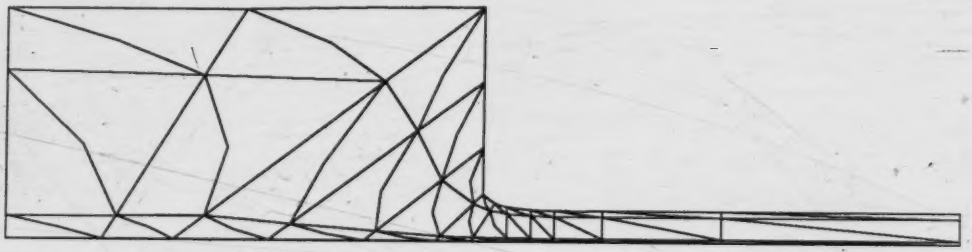


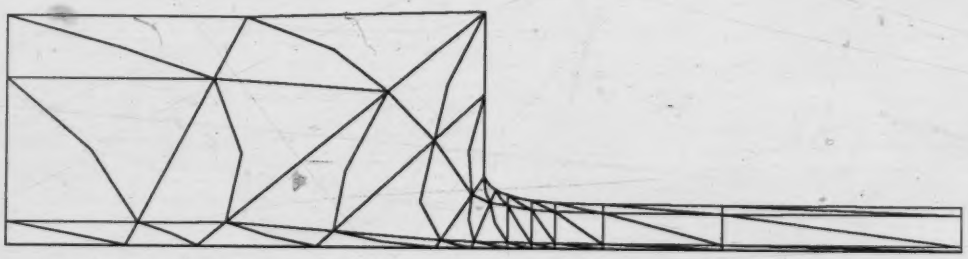
Fig. 37.- Contraction coefficients from Rajaratnam (1977) and from this study. Thickness of the bottom layer = 0.10 Y.



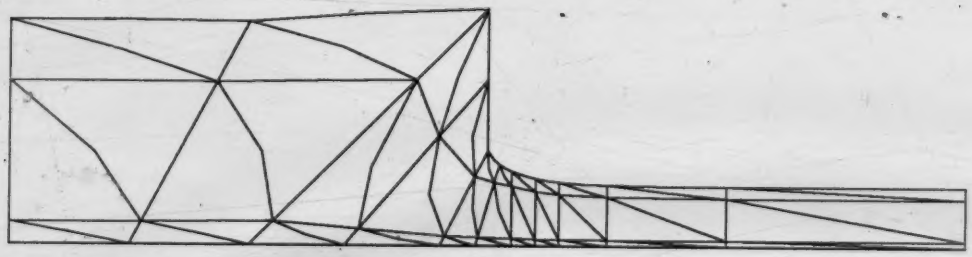
a/H = 0.1



a/H = 0.2



a/H = 0.3



a/H = 0.4

Fig. 38.- Meshes used to solve different gate opening-head ratios when vorticity is included.

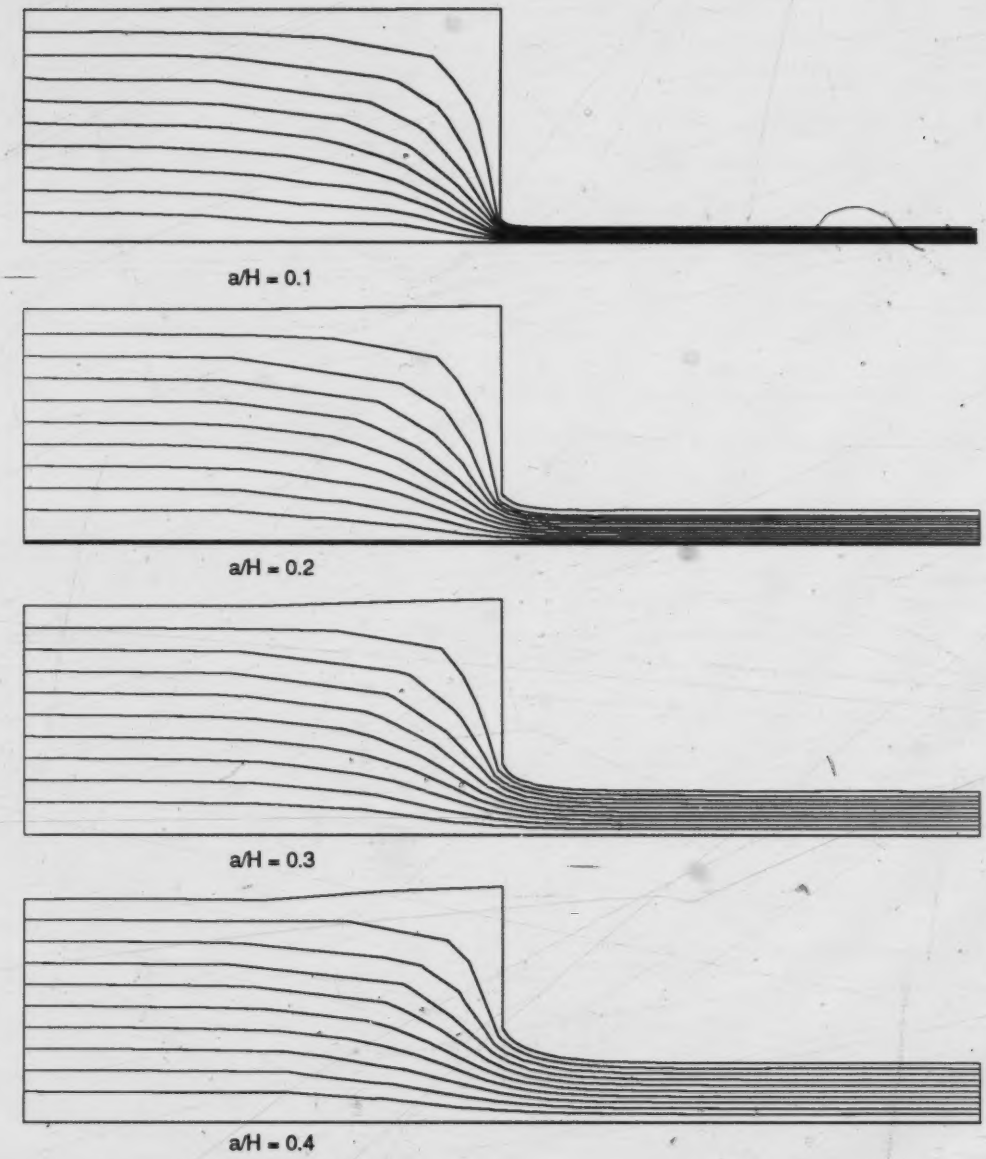
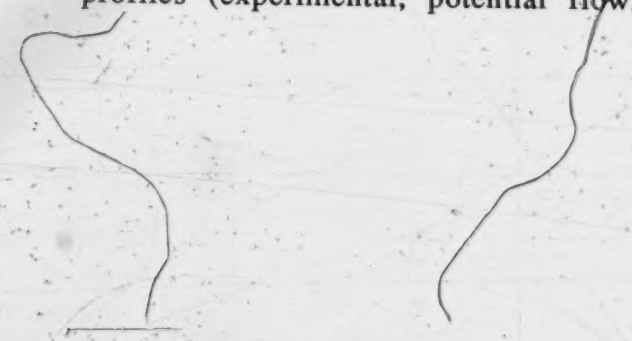


Fig. 39.- Streamlines for different gate opening-head ratios, solution obtained for a prescribed vorticity of -0.25 (du/dy).

Several thicknesses were tried in order to get a feeling of the influence of this. For a prescribed discharge, it was found that the thicker the layer the larger the contraction coefficient, this is because the influence of the vorticity extends over a larger domain, although its absolute value decreases. Figure 40 shows the effect of this with respect to the contraction coefficient.

Figures 41, 42, and 43 show the variation of the different coefficients as the thickness of the bottom layer increases. The discharge was not fixed, and the vorticity was prescribed.

Finally, two runs were made with actual results from the laboratory. The experiment is the one labelled A-IV in the publication by Rajaratnam (1977). One of the runs was done considering potential flow, and the second one with a prescribed vorticity of -0.25(du/dy). Figure 44 shows the three downstream profiles (experimental, potential flow, dynamic vorticity).



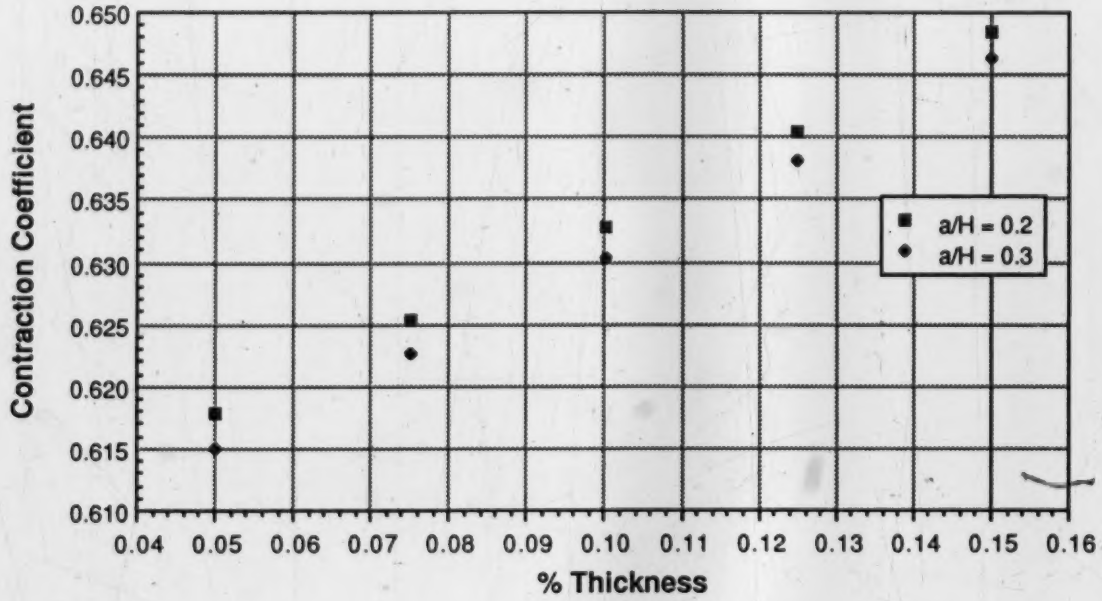


Fig. 40.- Effect of the thickness of the bottom layer on the contraction coefficient for a fixed discharge and a prescribed vorticity of -0.25(du/dy).

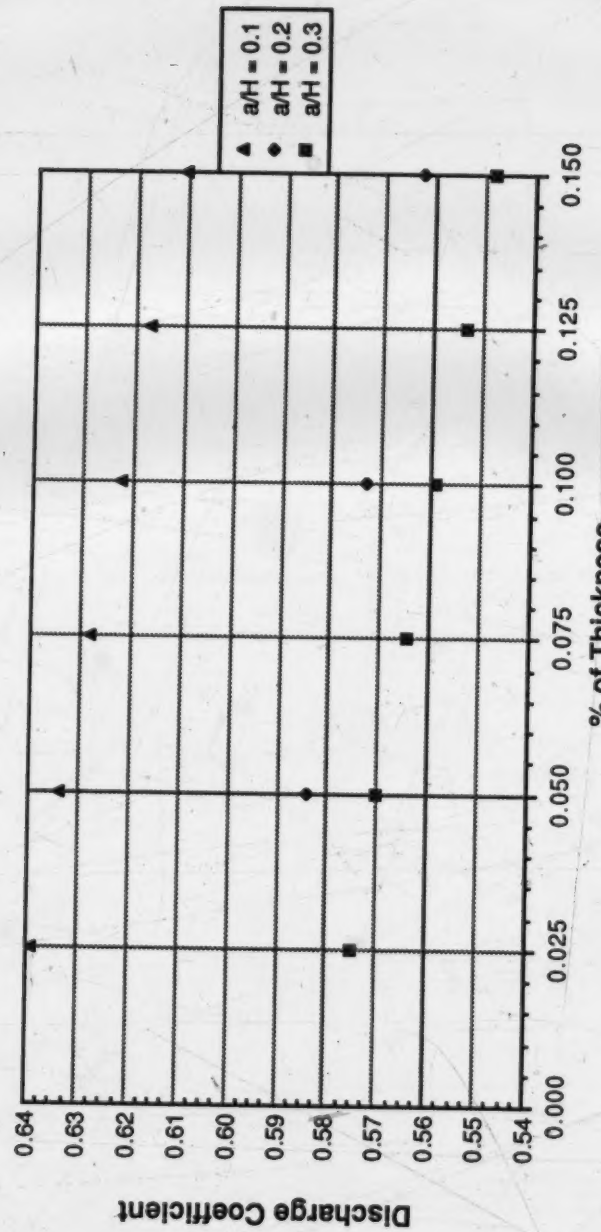


Fig. 41.- Effect of the thickness of the bottom layer on the discharge coefficient for a prescribed vorticity of -0.25(du/dy).

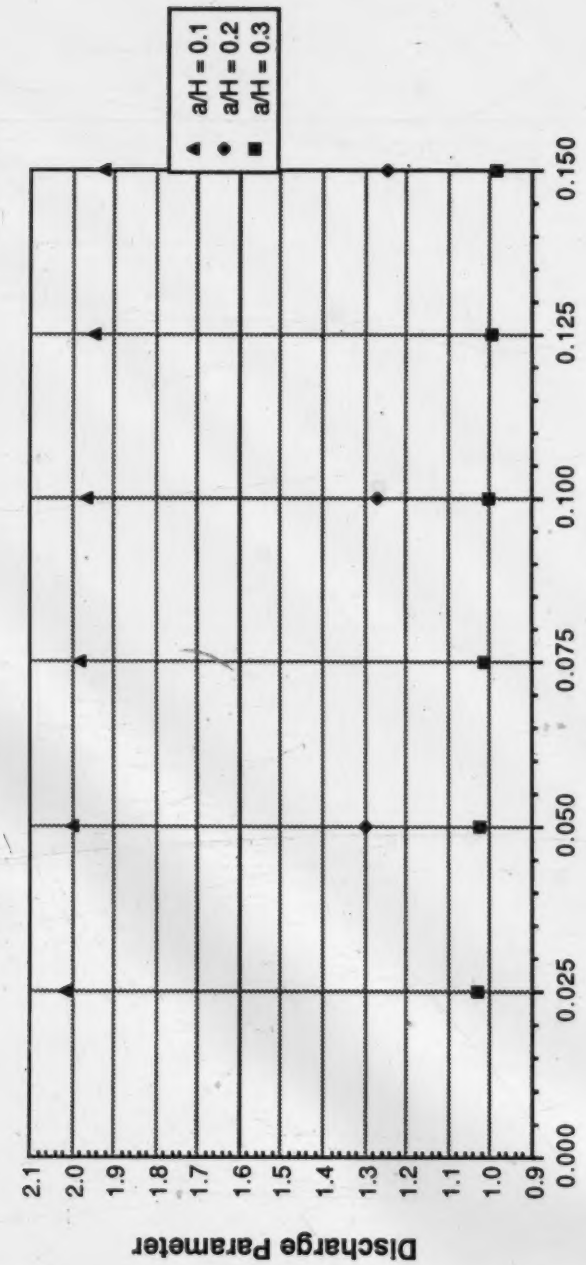


Fig. 42.- Effect of the thickness of the bottom layer on the discharge parameter for a prescribed vorticity of -0.25(du/dy).

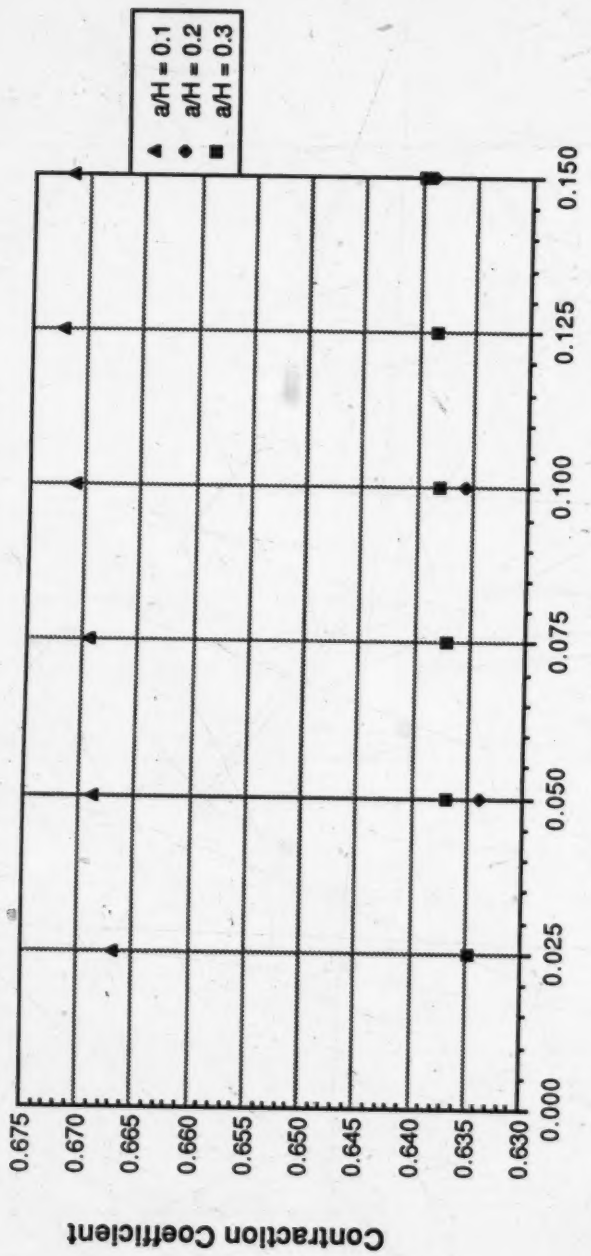


Fig. 43.- Effect of the thickness of the bottom layer on the contraction coefficient for a prescribed vorticity of -0.25(du/dy).

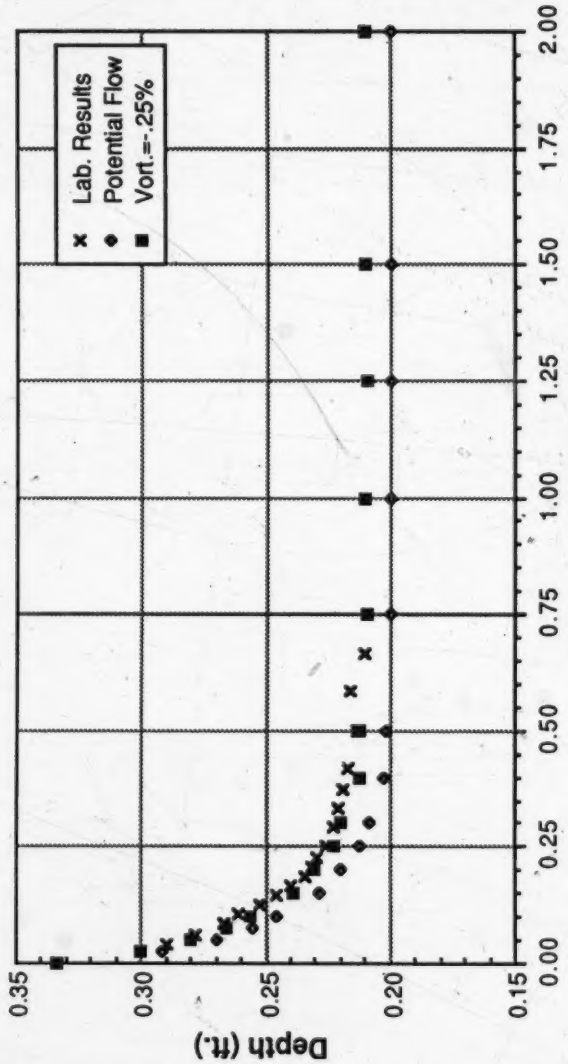


Fig. 44.- Comparison of numerical downstream free surface profiles with an experimental result from Rajaratnam (1977).

V.- CONCLUSIONS

A finite element method to solve the flow under sluice gates has been presented.

By using Poisson equation instead of Laplace equation to model the flow, higher contraction and discharge coefficients have been obtained, as well as the discharge parameter. The changes were greater in the contracction coefficients, so if the discharge is the main unknown and the downstream free surface profile is not needed, then a potential flow model is accurate enough to get results. On the other hand, if the free surface profile is the main concern then it is more accurate to use the vorticity model. It was found that if the discharge is fixed, there is an upper limit for the amount of vorticity that can be introduced to the model, and that the domain over where it is applied has an important influence in the overall results.

If the discharge is also iterated and vorticity introduced, there is a sudden change from the potential flow solution, but then it does not make a big difference wich amount of vorticity is used, or the thickness of the bottom layer. The flow adjusts to the new circimstances. Further research is needed to find which are the correct values to use, that means to calibrate the model.

After analyzing the experimental results it is proposed a new way to describe the relation between the gate opening-head ratio and the discharge, this is the discharge parameter as defined in chapter IV.

There are some special features in this model that make it easy to use and appropriate to implement in a microcomputer: the mesh generator is capable of producing graded meshes. The density of the grid is controlled by means of patched master elements and it can produce different kind of elements. Since in this problem the domain changes after each iteration it was needed that the mesh generator could be capable of regenerating the whole mesh without losing the relative proportion between elements. This was possible using super master elements. The time used to generate the mesh is about 25% of the total CPU. time.

As mentioned before, the calculation of the boundary velocities is very important. By using the algorithm proposed by Carey et al. very accurate boundary velocities are determined. This helps the free surface correction algorithm to get good values, this is because the algorithm used to update the free surface is based in the calculation of the difference in pressure at those nodes, to get this ΔP the velocity head is needed. An inaccurate velocity would give wrong corrections and since the flow downstream of the gate is supercritical a wrong velocity has a great influence; it is not the same upstream of the gate where the velocity is relatively small. The way to update the free surface is such that with a considerably less number of degrees of freedom good results are obtained. This feature is the one that makes the program accessible to a microcomputer since the memory requirements are proportional to the number of unknowns.

The implementation of this model to other flow problems seems to be quite direct although the boundary subroutines would have to be changed in order to satisfy the particular boundary

conditions for each problem. In the case of implementing this model to solve spillway problems some features would have to be added to the free surface update subroutine, because now it just works with a horizontal bed.

The model as it is may also be applied to inclined gates as well as radial gates, it would be a matter of using the correct data file.

LIST OF REFERENCES

Addison Herbert, "Supplementary Notes on Flow Through Model Sluices", Proc. Inst. Civ. Eng., Vol. 8, 53-72, 1937.

Benjamin B.J., "On the Flow in Channels when Rigid Obstacles are Placed in the Stream",J. of Fluid Mechs.,Vol. 1, 227-248, 1956.

Bettess P., BettessJ., "Analysis of Free Syrface Flows Using Isoparametric Finite Elements", Int. J. Num. Meth. Eng.,Vol. 19, 1675-1689, 1983.

Brebbia C.A., Wrobel L.C., "Applications of Boundary Elements in FluidFlow", Proc. 2nd. Intl. Conf. Finite Elements in Wat. Res., 4.67-4.85, London 1978.

Carey G.F., "Derivative Calculations from Finite Element Solutions", Comp. Meth. Applied Mechs. Eng., 35, 1-14, 1982.

Carey G.F., Chow S.S., Seager M.K., "Approximate Boundary Flux Calculations",Comp. Meth. Applied Mechs. Eng., 50, 107-120, 1985.

Chan T.K., Larock B.E., Herrmann L.R., "Free Surface Ideal Fluid Flows by Finite Elements", J. Hyd. Div., ASCE, Vol 99 (HY6), 959-974, 1973.

Chow V.T., Open Channel Hydraulics, McGraw Hill International Book Company, 1959.

Chung Y.K.," Solution of Flow Under Sluice Gates", J. Mechs. Div., ASCE, Vol. 98 (EM1), 121-140, 1972.

Diersch H.J., Martin H., "The Numerical Treatment of Free Surface Flows by Finite Elements",Proc. 2nd. Intl. Conf. Finite Elements in Wat. Res., 3.97-3.115, London 1978.

Diersch H.J., Schirmer A., Busch K.F., "Analysis of Flow with Initially Unknown Discharge", J. Hyd. Div., ASCE, Vol. 103 (HY3), 213-232, 1977.

Durocher L.L., Gasper A., "A Versatile Two-Dimensional Mesh Generator with Automatic Bandwidth Reduction", Computers and Structures, Vol. 10, 561-575, 1979.

Fangmeier D.D., Strelkoff T.S., "Solution for Gravity Flow Under a Sluice Gate",J. Mechs. Div., ASCE, Vol. 94 (EM1), 153-176, 1968.

Gray W.G., "Do Finite Element Models Simulate Surface Flow?", Proc. 3rd. Intl. Conf. Finite Elements in Wat. Res., 1.122-1.136, Mississipi 1980.

Gresho P.M., Lee R.L., "Don't Supress the Wiggles-They're Telling You Something!", Computers and Fluids, Vol. 9, 223-253, 1981.

Henderson F.M., Open Channel Flow, McMillan Company, 1966.

Heng F.L., Mitsoulis E., Prinos P., "Modelling of Flow Under Sluice Gates with FEM", Proc. VI Intl. Conf. Finite Elements in Wat. Res., 677-686, Lisboa 1986.

Henry H.R., "Discussion on Submerged Jets", Transactions ASCE, Vol. 115, 687-694, 1950.

Hurst H.E., Fraser W.D.A., "The Similarity of Motion of Water Through Sluices and Scale Models: Experiments with Models of Sluices of the Assuan Dam", Minutes Proc. Inst. Civ. Eng., Vol. 218, 72-112, 1923.

Isaacs L.T., "A Curved Cubic Triangular Finite Element for Potential Flow Problems", Int. J. Num. Meth. Eng., Vol. 7, 337-344, 1973.

Isaacs L.T., "Numerical Solution for Flow Under Sluice Gates", J. Hyd. Div., ASCE, Vol. 103 (HY5), 473-481, 1977.

Klassen V.J., "Flow from a Sluice Gate Under Gravity", J. Math. Anal. Appl., Vol. 19, 253-262, 1967.

Larock B.E., "Gravity-Affected Flow from Planar Sluice Gates", J. Hyd. Div., ASCE, Vol. 95 (HY4), 1211-1226, 1969.

Larock B.E., " A Theory for Free Outflow Beneath Radial Gates", J. Fluid Mechs., Vol. 41, 851-864, 1970.

Larock B.E., "Flow Over Gated Spillway Crests", Developments in Mechs., Vol. 8, 437-451, 1975.

Luke J.C., "A Variational Principle for a Fluid with a Free Surface", J. Fluid Mechs., Vol. 27, 395-397, 1967.

Martin C.H., Carey F.G., Introduction to Finite Element Analysis, McGraw Hill Book Company, 1973.

Masliyah J.H., Nandakumar K., Hemphill F., Fung L., "Body-Fitted Coordinates for Flow Under Sluice Gates", J. Hyd. Div., ASCE, Vol. 111, 922-933, 1985.

McCorquodale J.A., Li C.Y., "Finite Element Analysis of Sluice Gate Flow", Trans. Eng. Inst. Canada, Vol. 14 No. C-2, 1971

Pajer G., "Ueber der Stromungsvorgans an Einer Unterstromten Scharfkantigen Plaschutze", Zeit. Ang. Mat. Mec., Vol. 17, 259-269, 1937.

Rajaratnam N., "Free Flow Immediately Below Sluice Gates", J. Hyd. Div., ASCE, Vol. 103 (HY4), 345-351, 1977.

Rajaratnam N., Humphries J.A., "Free Flow Upstream of Vertical Sluice Gates", J. Hyd. Res., 20, 427-437, 1982.

Rajaratnam N., Subramanya K., "Flow Equation for the Sluice Gate", J. Irrigation and Drainage Div., ASCE, Vol. 93 (IR3), 167-186, 1967.

Räsänen S.M., Salonen E-M., "A Model for Understanding Free Surface Flow", Numerical Methods for Non Linear Problems, Vol.2, 849-860 Pineridge Press, 1984.

Sewell G., Analysis of a Finite Element Method, Springer-Verlag, 1985.

Stasa F., Applied Finite Element Analysis for Engineers, Holt Rinehart and Winston, 1985.

Strelkoff T.S., "Solution of Highly Curvilinear Gravity Flows", J. Mechs. Div., ASCE, Vol. 90 (EM3), 195-221, 1964.

Strelkoff T.S., Moayeri M.S., "Pattern of Potential Flow in a Free Overfall", J. Hyd. Div., ASCE, Vol. 96 (HY4), 879-901, 1970.

Thompson J.F., Warsi Z.U.A., Mastin C.W., Numerical Grid Generation, Foundations and Applications, North-Holland, 1985

Wilson D.C., "Smoothing Patched Grids", Computers and Fluids, Vol. 14, 11-22, 1986.

Zienkiewicz O.C., The Finite Element Method, 3rd. edition, McGraw Hill Book Company, 1977.

APPENDIX A

Program Listing

```
C
C     ************************************************
C     *              ALBERTO ALCARAZ                 *
C     *        DEPARTMENT OF CIVIL ENGINEERING       *
C     *           UNIVERSITY OF ALBERTA              *
C     *                ID  165970                    *
C     *               MSc. THESIS                    *
C     ************************************************
C
C
      INTEGER ELEM(1500,9),PSMAS,JDIAG(3000),PARAM,EL(1500,9),EXAM
     INTEGER NX(50),NY(50),NEX(50),NEY(50),AVGFLG,VORFLG,POINT(101)
     INTEGER SET,NELX(65),NELY(65),CIRFLG,NXX(50),NYY(50),NEXX(50),
     %        NEYY(50)
C
C
      REAL COORL(3000,2),COORD(3000,2),X(50,8),Y(50,8),KEL(9,9)
       REAL FEL(9),A(50000),C(50000),B(3000),A1(50000),R(300)
       REAL F1(3000),F(300),COR(3000,2),DPNOD(75),YY(50,8),VAL2(300)
      REAL JDIAG1(300),VORT(1500),VORTIC(50),SUPX(20,5),SUPY(20,5)
      REAL PCT(20,10,2),DNDR(16,9),DNDS(16,9)
C
C
     LOGICAL AFAC,BACK
C
C
C
C
C
     CALL PARDEF (NMASX,NMASY,NEBCR,PARAM,EXAM,HEAD,SSS,ITER,MOOTH,
     $TOL,NEBG,ALFA1,VORFLG,AVGFLG,LUMFLG,NPE,NPME,NMAS,NX,NY,VAL1,
     $VAL2,VAL3,NEX,NEY,NNX,NNY,NNODE,NELM,VORT,VORTIC,TOLQ,ITERQ,
    $SET,SUPX,SUPY,NELX,NELY,NELXD,NELYD,CIRFLG,PCT,VORIND,VORCIN,
    $KOUT,TOLEND,NXX,NYY,NEXX,NEYY,NNXX,NNYY,NELM2,NNODE2)
     ERRQ=1.
     KONTQ=1
     K=0
     ERRNOD=1.
     DPN=1.
     FLG=0.
15   M1=1
     CALL SUPMAS (SUPX,SUPY,PCT,X,Y,NMASX,NMASY)
     CALL MESH(NX,NY,NMASX,NMASY,NMAS,X,Y,SHAPE,COORD,ELEM,NELM,
    *NNODE,NNY,PARAM,NEX,NNX,MOOTH,NEY)
      IF(K.EQ.0)THEN
          CALL ENC (NMASX,NMASY,NMAS,NX,NY,ITER,PARAM,MOOTH,NELM,
     $     NNODE,ALFA1,TOL,VAL3,AVGFLG,LUMFLG,VORIND,VORCIN)
     ENDIF
C
C
C
     CALL ASEMBL (ELEM,COORD,NELM,NNODE,NNY,A,B,C,JDIAG,
      * A1,F1,M1,NPE,VAL1,VAL2,NUM4,EXAM,PARAM,NNX,VORT,FLG,SET,
    * DNDR,DNDS)
```

```
      CALL UACTCL (A,C,B,JDIAG,NNODE,AFAC,BACK)
      IF (VORFLG.EQ.1) THEN
          CALL DINVOR(NNX,NNY,COORD,B,PARAM,CIRFLG,NEBG,NEBCR,NELX,
     &       NELY,NELXD,NELYD,VORT,NEX,NMASY,VORIND,VORCIN,KOUT,
     &       KELVOR,KELVC1,KELVC2)
      ENDIF
      IF ((K.EQ.0.OR.K.EQ.ITER).AND.(ERRQ.LE.TOLQ.OR.KONTQ.EQ.
     &ITERQ)) THEN
          CALL OUTMAC (NNODE,NELM,COORD,ELEM,B,K,NX,NY,NEBCR,DPN,
     $    TOL,PARAM,HEAD,VAL3)
      ENDIF
C
C
C
      CALL SURF (A1,B,R,NNODE,ELEM,NNY,JDIAG,F1,NEX,NMASX,EL,
     *NELM,NOD,NEL,NPEL,NNX,COORD,COR,NEBCR,NUM4,NUM3,EXAM,NEBG,
     *NMASY,PARAM,VAL3,HEAD,FLG)
      M1=2
      CALL ASEMBL(EL,COR,NEL,NOD,NNY,A,B,C,JDIAG,A1,F1,M1,
     &          NPEL,VAL1,VAL2,NUM4,EXAM,PARAM,NNX,VORT,FLG,SET,
     &          DNDR,DNDS)
      IF (LUMFLG.EQ.0)THEN
          CALL UACTCL (A,C,R,JDIAG,NOD,AFAC,BACK)
      ELSEIF (LUMFLG.EQ.1)THEN
          CALL SOLVE (A,R,JDIAG,NOD,NUM4,EXAM,VAL3,COORD)
      ENDIF
C
C
C
      K=K+1
  108 FORMAT(//,4X,'FINAL SOLUTION AFTER ',I3,' ITERATIONS.')
  109 FORMAT (//,4X,'TOLERANCE OF ',F7.5,' NOT MET AFTER ',I3,
     &' ITERATIONS',/,4X'DPN IN LAST ITERATION = ',F9.5)
C
C
C
      IF(K.LE.ITER.AND.DPN.GT.TOL) THEN
          CALL NEWCOR (NX,NMASX,COR,X,Y,R,K,NNY,COORD,VAL3,NEBCR,
     &    ERRNOD,EXAM,HEAD,SSS,NEBG,ALFA1,NMASY,NOD,NEX,NUM4,PARAM,
     &    AVGFLG,DPNOD,POINT,NPOINT,ITER,SUPX,SUPY,KOUT)
          DPN=DPNOD(K)
          IF(K.GE.2)CALL RELAX (DPNOD,ALFA1,K,X,Y,YY,NEBCR,NMASY,
     &                    NMAS,NPME)
          IF(DPN.LE.TOL)GO TO 16
          GO TO 15
C
C
C
      ELSEIF (DPN.LE.TOL) THEN
   16     CALL OUT (COR,R,POINT,NPOINT,K,TOL,DPN,HEAD,DPNOD,VORFLG,
     $              CIRFLG,VORT,KELVOR,KELVC1,KELVC2)
          M1=1
          CALL MESH(NXX,NYY,NMASX,NMASY,NMAS,X,Y,SHAPE,COORD,ELEM,
     $    NELM2,NNODE2,NNYY,PARAM,NEXX,NNXX,MOOTH,NEYY)
```

```
      NUM3=0
      DO 241 LL=1,NEBCR
  241 NUM3=NUM3+NEXX(NMASY*LL)-1
      NUM3=(NUM3+1)*NNYY
      CALL ASEMBL (ELEM,COORD,NELM2,NNODE2,NNYY,A,B,C,JDIAG,
     *  A1,F1,M1,NPE,VAL1,VAL2,NUM4,EXAM,PARAM,NNXX,VORT,FLG,
     *  SET,DNDR,DNDS)
      CALL UACTCL (A,C,B,JDIAG,NNODE2,AFAC,BACK)
      IF(ERRQ.LE.TOLQ.OR.KONTQ.EQ.ITERQ)THEN
      CALL OUTMAC (NNODE,NELM,COORD,ELEM,B,K,NX,NY,NEBCR,
     $  DPN,TOL,PARAM,HEAD,VAL3,NNY)
      ENDIF
      CALL DIS(NUM3,NNYY,COORD,DNDR,DNDS,B,VG)
      IF (ERRQ.GT.TOLQ.AND.KONTQ.LE.ITERQ)THEN
        CALL QCHAN (COORD,VAL3,KONTQ,HEAD,ERRQ,VG,NUM3)
        ALFA1=0.1
        K=0
        TOL=TOL/2.
        IF (TOL.LT.TOLEND) TOL=TOLEND
        DPN=1.
        DO 119 I=1,NNX
  119   VAL2(I)=VAL3
        DO 139 I=1,NELM
  139   VORT(I)=0.
        GO TO 15
      ELSEIF (ERRQ.LE.TOLQ)THEN
        WRITE(6,120) KONTQ,VAL3
  120   FORMAT(//,'FINAL DISCHARGE MET AFTER ',I3,' ITERATIO',
     $  'NS     Q= ',F9.5)
        GO TO 888
      ELSEIF (ERRQ.GT.TOLQ) THEN
        WRITE(6,121)KONTQ,VAL3
  121   FORMAT(//,'TOLERANCE FOR THE DISCHARGE NOT MET AFTER',
     $  ' ',I3,' ITERATIONS, LAST Q= ',F9.5)
        GO TO 888
      ENDIF
    ELSEIF (K.GT.ITER) THEN
        WRITE(6,109)TOL,K,DPN
    ENDIF
  888 STOP
    END
C
C
C ***********************************************************************
C **    SUBROUTINE PARDEF-.IS THE INPUT SUBROUTINE AND ALSO          **
C **   DEFINES THE DIFFERENT PARAMETERS USED IN MESH AND             **
C **   THE VALUE OF THE VORTICITY FOR EACH ELEMENT.                  **
C ***********************************************************************
C
    SUBROUTINE PARDEF (NMASX,NMASY,NEBCR,PARAM,EXAM,HEAD,SSS,
    $ITER,MOOTH,TOL,NEBG,ALFA1,VORFLG,AVGFLG,LUMFLG,NPE,NPME,
    $NMAS,NX,NY,VAL1,VAL2,VAL3,NEX,NEY,NNX,NNY,NNODE,
   $NELM,VORT,VORTIC,TOLQ,ITERQ,SET,SUPX,SUPY,NELX,NELY,NELXD,
   $NELYD,CIRFLG,PCT,VORIND,VORCIN,KOUT,TOLEND,NXX,NYY,NEXX,NEYY,
```

```
     $NNXX,NNYY,NELM2,NNODE2)
      INTEGER PARAM,EXAM,NX(50),NY(50),NEX(50),NEY(50),AVGFLG,VORFLG
       REAL  VAL2(300),VORT(1500),VORTIC(50),PCT(20,10,2)
      REAL SUPX(20,5),SUPY(20,5)
      INTEGER NELX(50),NELY(50),NELT(50),SET,CIRFLG,NXX(50),NYY(50),
     &         NEXX(50),NEYY(50)
C
C          PARAM ---  PARAMETER THAT TELLS IF IT IS A QUADRATIC
C                     OR A LINEAR APPROXIMATION (1-LINEAR,2-QUAD)
C          EXAM ----- PARAMETER THAT TELLS THE KIND OF EXAMPLE,
C                     1-SUBCRITICAL CHANNEL,2-SUPERCRITICAL CHANNEL
C                     3-SLUICE GATE, 4-SPILLWAY
C          HEAD ----- THE INITIAL DEPTH UPSTREAM.
C          SSS    ----- SLOPE IN THE CHANNEL (FOR CHANNELS AND SLUICE)
C          ITER------ NUMBER OF ITERATIONS
C          MOOTH ---- NUMBER OF SMOOTHING DESIRED (0= NO SMOOTHING)
C          NEBG ----- NUMBER OF ELEMENTS BEFORE THE SLUICE GATE.
C           ALFA1 ----- RELAXATION FACTOR.
C          AVGFLG --- IF 1, USES AVERAGE VELOCITIES.
C          LUMFLG --- IF 1, USES LUMPED MATRIX FOR THE VELOCITIES.
C          VORFLG --- IF 1, USES DINAMYC VORTICITY IN THE ELEMENTS.
C
C
C
C
      READ (4,*)NMASX,NMASY,NEBCR,PARAM,EXAM,HEAD,SSS,ITER,MOOTH,
     &            TOL,TOLEND
       IF (EXAM.EQ.3) THEN
       READ(4,*)NEBG,ALFA1,VORFLG,AVGFLG,LUMFLG,CIRFLG,VORIND,VORCIN
      ELSE
           NEBG=NEBCR
            ALFA1=1.
           AVGFLG=0
           LUMFLG=0
           VORFLG=0
           CIRFLG=0
      ENDIF
       IF (PARAM.EQ.1) NPE=3
       IF (PARAM.EQ.2) NPE=6
       IF (PARAM.EQ.1) NPME=4
       IF (PARAM.EQ.2.OR.PARAM.EQ.3) NPME=8
       IF (PARAM.EQ.3) NPE=9
        IF (PARAM.NE.1.AND.PARAM.NE.2.AND.PARAM.NE.3)THEN
          WRITE (6,*) 'WRONG PARAMETER = ',PARAM
          GO TO 888
       ENDIF
      NMAS=NMASX*NMASY
       READ (4,*) (NX(I),I=1,NMAS,NMASY)
       READ (4,*) (NY(I),I=1,NMASY)
       READ (4,*) (NXX(I),I=1,NMAS,NMASY)
       READ (4,*) (NYY(I),I=1,NMASY)
       DO 4 I=1,NMAS,NMASY
           DO 4 J=1,NMASY-1
             NXX(I+J)=NXX(I)
```

```
 4   NX(I+J)=NX(I)
     DO 6 J=1,NMASY
          DO 6 I=1,NMASX
            NYY((I-1)*NMASY+J)=NYY(J)
 6   NY((I-1)*NMASY+J)=NY(J)
     DO 10 I=1,NMASX
 10   READ (4,*) (SUPX(I,J),SUPY(I,J),J=1,5),(PCT(I,J,1),PCT(I,J,2),
    &J=1,NMASY+1)
      READ (4,*) VAL1,VAL3,TOLQ,ITERQ,SET,KOUT
C
C
C    FIXING THE SUBCRITICAL DEPTH (UPSTREAM DEPTH)
C
     DIFER=1.
 15  IF(DIFER.LE.0.000009)GO TO 20
     VALO=SUPY(1,4)
      SUPY(1,4)=HEAD-VAL3**2/(64.4*VALO**2)
     DIFER=ABS(SUPY(1,4)-VALO)
     GO TO 15
C
C   DEFINITION OF PARAMETERS THAT GO INTO MESH
C
C  NMASY = NUMBER OF MASTER ELEMENTS IN THE Y DIRECTION.
C  NMAS = NUMBER OF MASTER ELEMENTS.
C  NNX = TOTAL NUMBER OF NODES IN THE X DIRECTION.
C  NNY = TOTAL NUMBER OF NODES IN THE Y DIRECTION.
C  NNODE = TOTAL NUMBER OF NODES.
C  NELM = TOTAL NUMBER OF ELEMENTS.
C  COORD = ARRAY THAT KEEPS NODAL GLOBAL COORDINATES.
C  ELEM = ARRAY THAT KEEPS NODES PER ELEMENT.
C  NEX(I)= ARRAY THAT KEEPS NUMBER OF DIVISION PER MASTER ELEMENT.
C  NEY(I)= ARRAY THAT KEEPS NUMBER OF DIVISION PER MASTER ELEMENT.
C
 20  IF (PARAM.EQ.1)THEN
         DO 25 I=1,NMAS
           NEX(I)=NX(I)
 25      NEY(I)=NY(I)
     ELSE
         DO 30 I=1,NMAS
           NEXX(I)=2*NXX(I)-1
           NEYY(I)=2*NYY(I)-1
          NEX(I)=2*NX(I)-1
 30      NEY(I)=2*NY(I)-1
     ENDIF
     NNX=0
     NNY=0
     NNXX=0
     NNYY=0
     NUMX=NMAS/NMASY+1
     NUMY=NMAS/NMASX+1
     DO 35 I=1,NMAS,NMASY
          NNXX=NNXX+NEXX(I)-1
 35  NNX=NNX+NEX(I)-1
     DO 40 I=1,NMASY
```

```
          NNYY=NNYY+NEYY(I)-1
   40  NNY=NNY+NEY(I)-1
       NNX=NNX+1
       NNY=NNY+1
       NNXX=NNXX+1
       NNYY=NNYY+1
       NELM=0
       NELM2=0
        IF(PARAM.EQ.1.OR.PARAM.EQ.2) THEN
            DO 45 I=1,NMAS
              NELM2=NELM2+2*(NXX(I)-1)*(NYY(I)-1)
   45       NELM=NELM+2*(NX(I)-1)*(NY(I)-1)
        ELSEIF (PARAM.EQ.3) THEN
            DO 46 I=1,NMAS
   46       NELM=NELM+(NX(I)-1)*(NY(I)-1)
        ENDIF
       NNODE=NNX*NNY
       NNODE2=NNXX*NNYY
        IF(PARAM.EQ.2.OR.PARAM.EQ.3)THEN
            NELXD=(NNX-1)/2
            NELYD=(NNY-1)/2
        ELSEIF(PARAM.EQ.1)THEN
            NELXD=NNX-1
            NELYD=NNY-1
        ENDIF
        DO 55 I=1,NNX
   55   VAL2(I)=VAL3
C
C   ASSIGNING VORTICITY TO THE DIFFERENT ELEMENTS
C
        IF(PARAM.EQ.2) THEN
            DO 60 I=1,NMAS
               NELX(I)=NX(I)-1
               NELY(I)=NY(I)-1
   60       NELT(I)=NELX(I)*NELY(I)*2
        ENDIF
        IF (PARAM.EQ.3) THEN
            DO 61 I=1,NMAS
               NELX(I)=NX(I)-1
               NELY(I)=NY(I)-1
   61       NELT(I)=NELX(I)*NELY(I)
        ENDIF
        DO 98 I=1,NELM
   98   VORT(I)=0.0
  888 RETURN
        END
C
C
C*************************************************************************
C **  SUBROUTINE SUPMASTER-. IT CREATES THE MASTER ELEMENTS THAT     **
C **  GO INTO "MESH", IT JUST REFINES THE ELEMENTS ALONG THE         **
C **  "Y" DIRECTION.                                                 **
C*************************************************************************
C
```

```
C
      SUBROUTINE SUPMAS (SUPX,SUPY,PCT,X,Y,NMASX,NMASY)
C
C
          REAL  X(50,8),Y(50,8),SUPX(20,5),SUPY(20,5),PCT(20,10,2)
C
C
       DO 20 I=1,NMASX
            DO 20 J=1,NMASY
              KT1=J+(I-1)*NMASY
               Y(KT1,1)=PCT(I,J,1)*SUPY(I,4)+SUPY(I,1)
                X(KT1,1)=((SUPX(I,4)-SUPX(I,1))*Y(KT1,1))/(SUPY(I,4)-
     &                  SUPY(I,1))+SUPX(I,1)
               Y(KT1,2)=PCT(I,J,2)*SUPY(I,3)+SUPY(I,2)
                X(KT1,2)=((SUPX(I,3)-SUPX(I,2))*Y(KT1,2))/(SUPY(I,3)-
     &                  SUPY(I,2))+SUPX(I,2)
               Y(KT1,3)=PCT(I,J+1,2)*SUPY(I,3)+SUPY(I,2)
                X(KT1,3)=((SUPX(I,3)-SUPX(I,2))*Y(KT1,3))/(SUPY(I,3)-
     &                  SUPY(I,2))+SUPX(I,2)
               Y(KT1,4)=PCT(I,J+1,1)*SUPY(I,4)+SUPY(I,1)
                X(KT1,4)=((SUPX(I,4)-SUPX(I,1))*Y(KT1,4))/(SUPY(I,4)-
     &                  SUPY(I,1))+SUPX(I,1)
               X(KT1,5)=(X(KT1,1)+X(KT1,2))/2.
               Y(KT1,5)=(Y(KT1,1)+Y(KT1,2))/2.
               X(KT1,6)=(X(KT1,2)+X(KT1,3))/2.
               Y(KT1,6)=(Y(KT1,2)+Y(KT1,3))/2.
               X(KT1,8)=(X(KT1,1)+X(KT1,4))/2.
               Y(KT1,8)=(Y(KT1,1)+Y(KT1,4))/2.
              IF(J.NE.NMASY) THEN
                  X(KT1,7)=(X(KT1,3)+X(KT1,4))/2.
                  Y(KT1,7)=(Y(KT1,3)+Y(KT1,4))/2.
              ELSEIF(J.EQ.NMASY)THEN
                 X(KT1,7)=SUPX(I,5)
                 Y(KT1,7)=SUPY(I,5)
              ENDIF
 20  CONTINUE
     RETURN
     END
C
C
C ***********************************************************************
C ***     SUBROUTINE ENC-. PRINTS THE GENERAL INFORMATION          **
C ***     ABOUT THE PROBLEM.                                       **
C ***********************************************************************
C
C
      SUBROUTINE ENC (NMASX,NMASY,NMAS,NX,NY,ITER,PARAM,MOOTH,NELM,
     $NNODE,ALFA1,TOL,VAL3,AVGFLG,LUMFLG,VORIND,VORCIN)
C
      INTEGER NX(50),NY(50),PARAM,AVGFLG,LUMFLG
      REAL VORTIC (50)
C
C   INFORMATION ABOUT THE ELEMENTS
C
```

```
      WRITE (6,106)
      WRITE (6,101) (NX(I),I=NMASY,NMAS,NMASY)
  101 FORMAT('********** NX(I)= ',15(I2,','),'**********')
      WRITE(6,102)(NY(I),I=1,NMASY)
  102 FORMAT('********** NY(I)= ',3(I2,','),21X,'**********')
      WRITE (6,103) ITER,PARAM,MOOTH,,NELM,NNODE,ALFA1
  103 FORMAT('********** NUMBER OF ITERATIONS= ',I2,20X,'**********',
     &'*',/,'********** ELEMENTS TYPE= ',I1,' (1=LINEAR, 2=QUADRAT',
     &'IC)',4X,'**********'/,'********** NUMBER OF SMOOTHING= ',I1,
     &22X,'**********'/,'********** NUMBER OF ELEMENTS= ',I4,20X,
     &'**********'/,'********** NUMBER OF NODES= ',I4,23X,'*****',
     &'*****'/,'********** INITIAL RELAXATION FACTOR= ',F4.2,
     &13X,'**********')
      WRITE(6,108) VORIND,VORCIN
  108 FORMAT ('**********  RELAXATION FACTOR FOR THE VORTICITY ',
     &'      **********'/,'********** WITHIN THE ELEMENTS ALONG',
     &' THE BED = ',F7.6,'**********'/,'********** RELAXATION ',
     &'FACTOR FOR THE VORTICITY          **********'/,'********** ',
     &'WITHIN THE ELEMENTS ALONG THE GATE= ',F7.6,'**********')
      WRITE(6,110)TOL
  110 FORMAT('********** TOLERANCE= ',F7.5,26X,'**********')
      WRITE(6,111)VAL3
  111 FORMAT('********** ASSUMED DISCHARGE = ',F8.5,' CFS/FT',9X,
     &'**********')
      IF(AVGFLG.EQ.1)WRITE(6,104)
  104 FORMAT('********** USING AVERAGE VELOCITIES',20X,'**********')
      IF(LUMFLG.EQ.1)WRITE (6,105)
  105 FORMAT('********** USING LUMPED VALUES FOR THE VELOCITIES',
     &6X,'**********')
      WRITE(6,106)
  106 FORMAT('**************************************************',
     &'**************')
C
      RETURN
      END
C
C
C  ************************************
C  *     SHAPE FUNCTION          **
C  ************************************
C
C
C
      FUNCTION SHAPE (R,S,Z,L,PARAM)
      DIMENSION Z(50,8)
      INTEGER PARAM
C
      IF (PARAM.EQ.1)THEN
          S1=(1.-R)*(1.-S)/4.*Z(L,1)
          S2=(1.+R)*(1.-S)/4.*Z(L,2)
          S3=(1.+R)*(1.+S)/4.*Z(L,3)
          S4=(1.-R)*(1.+S)/4.*Z(L,4)
         SHAPE=S1+S2+S3+S4
      ELSEIF (PARAM.EQ.5) THEN
```

```
      S1=(1.-R-S)*(2.*(1.-R-S)-1.)*Z(L,1)
      S2=R*(2.*R-1.)*Z(L,2)
      S3=S*(2.*S-1.)*Z(L,3)
      S4=4.*R*(1.-R-S)*Z(L,4)
      S5=4.*R*S*Z(L,5)
      S6=4.*S*(1.-R-S)*Z(L,6)
      SHAPE=S1+S2+S3+S4+S5+S6
   ELSE
      S1=-(1.-R)*(1.-S)*(R+S+1.)/4.*Z(L,1)
      S2=(1.+R)*(1.-S)*(R-S-1.)/4.*Z(L,2)
      S3=(1.+R)*(1.+S)*(R+S-1.)/4.*Z(L,3)
      S4=(1.-R)*(1.+S)*(S-R-1.)/4.*Z(L,4)
      S5=(1.-R**2)*(1.-S)/2.*Z(L,5)
      S6=(1.+R)*(1.-S**2)/2.*Z(L,6)
      S7=(1.-R**2)*(1.+S)/2.*Z(L,7)
      S8=(1.-R)*(1.-S**2)/2.*Z(L,8)
```

```
C LOCAL COORDINATE GENERATION
C
C
      K=0
      DO 15 I=1,NMAS,NMASY
         DX=2./(NEX(I)-1)
         IF (I.EQ.1)THEN
            K3=1
         ELSE
            K3=2
         ENDIF
         UX=-1.
         DO 15 IX=K3,NEX(I)
          DO 15 J=1,NMASY
             DY=2./(NEY(J)-1)
             IF (J.EQ.1)THEN
                K2=1
            ELSE
                K2=2
            ENDIF
            UY=-1.
            DO 15 IY=K2,NEY(J)
                K=K+1
                COORL(K,1)=UX+DX*(IX-1)
                COORL(K,2)=UY+DY*(IY-1)
 15   CONTINUE
C
C   MAPPING TO GLOBAL COORDINATES FROM LOCAL COORDINATES
C
      K=0
      DO 18 I=1,NMAS,NMASY
          K1=I-1
          IF (I.EQ.1)THEN
             K3=1
          ELSE
             K3=2
          ENDIF
          DO 18 IX=K3,NEX(I)
            DO 18 J=1,NMASY
              L=J+K1
              IF (J.EQ.1) THEN
                 K2=1
             ELSE
                 K2=2
             ENDIF
             DO 18 IY=K2,NEY(J)
                K=K+1
                COORD(K,1)=SHAPE(COORL(K,1),COORL(K,2),X,L,PARAM)
                COORD(K,2)=SHAPE(COORL(K,1),COORL(K,2),Y,L,PARAM)
 18   CONTINUE
C
C    SMOOTHING PROCEDURE
C
      IF(MOOTH.NE.0)THEN
```

```
      DO 21 LL=1,MOOTH
      K1=NNX-2
      K2=NNY-2
      K3=NNY-1
      DO 21 I=1,K1
        K3=K3+2
        DO 21 J=1,K2
        K3=K3+1
       COORD(K3,1)=(COORD(K3+1,1)+COORD(K3-1,1)+COORD(K3+NNY,1)+
     *  COORD(K3-NNY,1))/4.
       COORD(K3,2)=(COORD(K3+1,2)+COORD(K3-1,2)+COORD(K3+NNY,2)+
     *  COORD(K3-NNY,2))/4.
 21    CONTINUE
   ENDIF
C
C
C  DEFINE NODES PER ELEMENT
C
C
   K6=1
   K5=1
   IF (PARAM.EQ.2)THEN
      DO 70 I=1,NELM,2
        ELEM(I,1)=K5
        ELEM(I,2)=K5+2*NNY
        ELEM(I,3)=K5+2
        ELEM(I,4)=K5+NNY
        ELEM(I,5)=K5+NNY+1
        ELEM(I,6)=K5+1
C
        ELEM(I+1,1)=K5+NNY*2+2
        ELEM(I+1,3)=K5+2*NNY
        ELEM(I+1,2)=K5+2
        ELEM(I+1,6)=K5+2*NNY+1
        ELEM(I+1,5)=K5+NNY+1
        ELEM(I+1,4)=K5+NNY+2
C
        K5=K5+2*NNY
        IF (MOD(K5,NMX1).EQ.0) THEN
          K5=K6+2
          K6=K6+2
         NMX1=NMX1+2
        ENDIF
 70   CONTINUE
   ELSEIF (PARAM.EQ.1)THEN
      DO 80 I=1,NELM,2
        ELEM(I,1)=K5
        ELEM(I,2)=K5+NNY
        ELEM(I,3)=K5+1
C
        ELEM(I+1,1)=K5+NNY+1
        ELEM(I+1,2)=K5+1
        ELEM(I+1,3)=K5+NNY
C
```

```
            K5=K5+NNY
            IF(MOD(K5,NMX1).EQ.0) THEN
                K5=K6+1
                K6=K6+1
                NMX1=NMX1+1
            ENDIF
 80       CONTINUE
        ELSEIF (PARAM.EQ.3)THEN
            DO 90 I=1,NELM
             ELEM(I,1)=K5
             ELEM(I,2)=K5+2*NNY
             ELEM(I,3)=K5+2*NNY+2
             ELEM(I,4)=K5+2
             ELEM(I,5)=K5+NNY
             ELEM(I,6)=K5+2*NNY+1
             ELEM(I,7)=K5+NNY+2
             ELEM(I,8)=K5+1
             ELEM(I,9)=K5+NNY+1
C
            K5=K5+2*NNY
            IF(MOD(K5,NMX1).EQ.0)THEN
                K5=K6+2
                K6=K6+2
                NMX1=NMX1+2
            ENDIF
 90       CONTINUE
        ENDIF
        RETURN
        END
C
C
C
C
C*************************************************************************
C *       SUBROUTINE TRELEM-. IS THE ELEMENT SUBROUTINE                 **
C *       FOR QUADRATIC TRIANGLES, TO INTEGRATE IT USES                 **
C *       NUMERICAL INTEGRATION TAKING THREE POINTS.                    **
C*************************************************************************
C
C
      SUBROUTINE TRELEM(ELEM,COORD,N,KEL,FEL,VORT,DNDR,DNDS,VI,NIPT)
C
        REAL  COORD(3000,2),X(6),Y(6),DNDR(16,9),DNDS(16,9),KEL(9,9),
         *FEL(9),A(3,6),B(3,6),DET(3),J11(3),J12(3),J21(3),J22(3),
       *G(3,6,6),VI(16,9),VORT(1500)
        INTEGER ELEM(1500,9)
C
C  DNDR,DNDS = ARRAYS THAT KEEP THE PARTIAL DERIVATIVES FOR THE
C              INTEGRATION POINTS.
C  JNN         =ARRAYS THAT KEEP THA JACOBIAN FOR THE DIFFERENT
C        INTEGRATION POINTS.
C  DET = ARRAY THAT KEEPS THE VALUES OF THE DETERMINANT.
C  KEL = ELEMENT "STIFFNES" MATRIX.
C  FEL = RIGHT HAND SIDE ELEMENT VECTOR
```

```
C
C
C
      DO 5 I=1,6
    5 FEL(I)=0.
      DO 10 I=1,6
         X(I)=COORD(ELEM(N,I),1)
         Y(I)=COORD(ELEM(N,I),2)
   10 CONTINUE
      DO 12 I=1,NIPT
         J11(I)=0.
         J12(I)=0.
         J21(I)=0
         J22(I)=0.
   12 CONTINUE
      DO 13 I=1,3
         DO 13 J=1,6
           J11(I)=J11(I)+DNDR(I,J)*X(J)
           J12(I)=J12(I)+DNDR(I,J)*Y(J)
           J21(I)=J21(I)+DNDS(I,J)*X(J)
           J22(I)=J22(I)+DNDS(I,J)*Y(J)
   13 CONTINUE
      DO 20 I=1,3
   20   DET(I)=(J11(I)*J22(I))-(J21(I)*J12(I))
      DO 30 I=1,3
         DO 30 J=1,6
            A(I,J)=(J22(I)*DNDR(I,J)-J12(I)*DNDS(I,J))/DET(I)
            B(I,J)=(J11(I)*DNDS(I,J)-J21(I)*DNDR(I,J))/DET(I)
   30 CONTINUE
      DO 40 K=1,3
         DO 40 I=1,6
           DO 40 J=1,6
            G(K,I,J)=A(K,I)*A(K,J)+B(K,I)*B(K,J)
   40 CONTINUE
      DO 50 I=1,6
         DO 50 J=1,6
   50    KEL(I,J)=(G(1,I,J)*DET(1)+G(2,I,J)*DET(2)+
     &G(3,I,J)*DET(3))/6.
      DO 70 I=1,6
         DO 70 J=1,3
   70  FEL(I)=FEL(I)+VI(J,I)*DET(J)
      DO 75 I=1,6
   75  FEL(I)=FEL(I)*VORT(N)
C        WRITE(7,78)N
C 78     FORMAT(//,' ELEMENT NUMBER ',I3,//)
C        DO 76 I=1,6
C 76      WRITE(7,77)  (KEL(I,J),J=1,6)
C 77      FORMAT (6(2X,F8.5))
      RETURN
      END
C
C
C ***********************************************
C *          SUBROUTINE ELEM2                   **
```

```
C ***************************************
C
C
      SUBROUTINE ELEM2 (ELEM,COORD,I,KEL,FEL)
        REAL  COORD(3000,2),KEL(9,9),FEL(9)
       INTEGER ELEM(1500,9)
       DO 10 II=1,6
           DO  10  JJ=1,6
   10   KEL(II,JJ)=0.
       X1=COORD(ELEM(I,1),1)
       Y1=COORD(ELEM(I,1),2)
       X2=COORD(ELEM(I,2),1)
       Y2=COORD(ELEM(I,2),2)
       X3=COORD(ELEM(I,3),1)
       Y3=COORD(ELEM(I,3),2)
         R=.932469514203152
         W=.171324492379170
      KT=1
   20  RAIZ=SQRT(((R-.5)*X1-2.*R*X2+(R+.5)*X3)**2+((R-.5)*Y1-2*R*Y2
     &        +(R+.5)*Y3)**2)
      F11=(R*R*R*R-2.*R*R*R+R*R)*RAIZ
      F12=(-(R*R*R*R)+R*R*R+R*R-R)*RAIZ
      F13=(R*R*R*R-R*R)*RAIZ
      F22=(R*R*R*R-2.*R*R+1.)*RAIZ
      F23=(-(R*R*R*R)-(R*R*R)+R*R+R)*RAIZ
     F33=(R*R*R*R+2.*R*R*R+R*R)*RAIZ
       KEL(1,1)=KEL(1,1)+W*F11
       KEL(1,2)=KEL(1,2)+W*F12
       KEL(1,3)=KEL(1,3)+W*F13
       KEL(2,2)=KEL(2,2)+W*F22
       KEL(2,3)=KEL(2,3)+W*F23
       KEL(3,3)=KEL(3,3)+W*F33
      KT=KT+1
      IF (KT.EQ.2)THEN
          R=-R
          GO TO 20
      ELSEIF (KT.EQ.3)THEN
             R=.661209386466265
             W=.360761573048139
           GO TO 20
      ELSEIF (KT.EQ.4) THEN
          R=-R
          GO TO 20
      ELSEIF (KT.EQ.5)THEN
             R=.238619186083197
             W=.467913934572691
           GO TO 20
      ELSEIF (KT.EQ.6)THEN
         R=-R
         GO TO 20
     ENDIF
       KEL(1,1)=KEL(1,1)/4.
       KEL(1,2)=KEL(1,2)/2.
       KEL(1,3)=KEL(1,3)/4.
```

```
      KEL(2,2)=KEL(2,2)
      KEL(2,3)=KEL(2,3)/2.
      KEL(3,3)=KEL(3,3)/4.
      KEL(2,1)=KEL(1,2)
      KEL(3,1)=KEL(1,3)
      KEL(3,2)=KEL(2,3)
      FEL(1)=0.
      FEL(2)=0.
      FEL(3)=0.
      RETURN
      END
C
C
C
C
C*****************************************************************
C **  SUBROUTINE RECTNG-. IT CREATES THE LOCAL MATRIX FOR     **
C **  RECTANGULAR QUADRATIC ELEMENTS WITH 9 NODES.            **
C*****************************************************************
C
C
      SUBROUTINE RECTNG(ELEM,COORD,N,KEL,FEL,VORT,SET,DNDR,DNDS,
     & VI,NIPT,WI,WJ)
C
        REAL  COORD(3000,2),X(9),Y(9),DNDR(16,9),DNDS(16,9),KEL(9,9),
       &FEL(9),A(16,9),B(16,9),DET(16),J11(16),J12(16),J21(16),
       &J22(16),G(16,9,9),VI(16,9),VORT(1500),R(16),S(16),WI(16),
      &WJ(16),MR,MS
       INTEGER ELEM(1500,9),SET
C
C     SET=1
      DO 5 I=1,9
    5  FEL(I)=0.
      DO 10 I=1,9
           X(I)=COORD(ELEM(N,I),1)
           Y(I)=COORD(ELEM(N,I),2)
   10  CONTINUE
      DO 26 I=1,NIPT
           J11(I)=0.
           J12(I)=0.
           J21(I)=0.
           J22(I)=0.
   26  CONTINUE
      DO 27 I=1,NIPT
           DO 27 J=1,9
             J11(I)=J11(I)+DNDR(I,J)*X(J)
             J12(I)=J12(I)+DNDR(I,J)*Y(J)
             J21(I)=J21(I)+DNDS(I,J)*X(J)
             J22(I)=J22(I)+DNDS(I,J)*Y(J)
   27  CONTINUE
      DO 28 I=1,NIPT
         DET(I)=(J11(I)*J22(I))-(J21(I)*J12(I))
   28  CONTINUE
      DO 30 I=1,NIPT
```

```
          DO 30 J=1,9
             A(I,J)=(J22(I)*DNDR(I,J)-J12(I)*DNDS(I,J))/DET(I)
             B(I,J)=(J11(I)*DNDS(I,J)-J21(I)*DNDR(I,J))/DET(I)
 30  CONTINUE
     DO 41 K=1,NIPT
          DO 40 I=1,9
            DO 40 J=1,9
             G(K,I,J)=A(K,I)*A(K,J)+B(K,I)*B(K,J)
 40  CONTINUE
 41   CONTINUE
     DO 50 I=1,9
          DO 50 J=1,9
            KEL(I,J)=0.
           DO 50 K=1,NIPT
              KEL(I,J)=KEL(I,J)+(G(K,I,J)*DET(K)*WI(K)*WJ(K))
 50  CONTINUE
     DO 70 I=1,9
          DO 70 J=1,NIPT
 70   FEL(I)=FEL(I)+VI(J,I)*DET(J)
     DO 75 I=1,9
 75  FEL(I)=FEL(I)*VORT(N)
     RETURN
     END
C
C
C
C ***************************************
C *        SUBROUTINE BNDRY        **
C ***************************************
C
C
      SUBROUTINE BNDRY (A,B,JDIAG,NNODE,NNY,F1,VAL1,VAL2)
      INTEGER BDY(4000),JDIAG(3000)
        REAL  A(50000),B(3000),F1(3000),VAL2(300)
C
C
C  A = VECTOR THAT KEEPS ALL THE VALUES OF THE STIFFNES MATRIX.
C  B = RIGHT HAND SIDE VECTOR.
C  JDIAG = VECTOR THAT KEEPS TRACK OF THE DIAGONAL IN THE SKYLINE
C          PROFILE.
C  BDY = VECTOR THAT KEEPS THE NODE NUMBERS OF THE BOUNDARY PTS.
C  VAL1= BOUNDARY VALUE AT THE BOTTOM.
C  VAL2= BOUNDARY VALUE AT THE TOP.
C
C
     K=NNODE-NNY+1
     KONT=1
     DO 10 I=1,K,NNY
          BDY(KONT)=I
           B(I)=VAL1*100000000.
          F1(I)=VAL1
 10  KONT=KONT+1
     KKK=0
     DO 20 I=NNY,NNODE,NNY
```

```
          KKK=KKK+1
          BDY(KONT)=I
           B(I)=VAL2(KKK)*100000000.
          F1(I)=VAL2(KKK)
 20  KONT=KONT+1
     KONT=KONT-1
     DO 30 I=1,KONT
          L=JDIAG(BDY(I))
 30    A(L)=A(L)+100000000.
     RETURN
     END
C
C

C ****************************************
C *          SUBROUTINE ASEMBL       *
C ****************************************
C
C
C
     SUBROUTINE ASEMBL (ELEM,COORD,NELM,NNODE,NNY,A,B,C,JDIAG,
     + A1,F1,M1,NPE,VAL1,VAL2,NUM4,EXAM,PARAM,NNX,VORT,FLG,SET,
    + DNDR,DNDS)
       INTEGER ELEM(1500,9),JDIAG(3000),HDIF(9)
      INTEGER HDIF1(3000),M(8,9),EXAM,PARAM,SET
        REAL  A(50000),B(3000),COORD(3000,2),KEL(9,9),VORT(1500)
        REAL  FEL(9),C(50000),A1(50000),F1(3000),KKK(100,100)
       REAL  DNDR(16,9),DNDS(19,9),VI(16,9),WI(16),WJ(16)
C
C
C          LOCATION OF THE DIAGONAL IN THE SKYLINE VECTOR
C
C
      IF(M1.EQ.1)KL1=2**31-1
     JDIAG(1)=1
     KT5=1
     DO 100 J=2,NNODE
          KT1=1
          KT5=KT5+1
          DO 200 I=1,NELM
            DO 300 K=1,NPE
              IF(ELEM(I,K).EQ.J)THEN
                   DO 400 K1=1,NPE
 400               HDIF(K1)=ELEM(I,K)-ELEM(I,K1)
                   HDIF1(KT1)=NIMAX(HDIF,NPE)
                   KT1=KT1+1
              ENDIF
 300        CONTINUE
 200       CONTINUE
          KT1=KT1-1
          LCOL=NIMAX(HDIF1,KT1)
 100  JDIAG(KT5)=JDIAG(KT5-1)+1+LCOL

C
```

```
C            ASSEMBLY OF THE GLOBAL MATRIX
C
     KK=JDIAG(NNODE)
     DO 415 I=1,KK
        A(I)=0.
        C(I)=0.
  415  A1(I)=0.
     DO 405 I=1,NNODE
          C(JDIAG(I))=1.
          F1(I)=0.
  405  B(I)=0.
     IF (M1.EQ.1) CALL BNDRY (A,B,JDIAG,NNODE,NNY,F1,VAL1,VAL2)
     IF (M1.EQ.1) CALL GAUSS (DNDR,DNDS,VI,PARAM,SET,NIPT,WI,WJ)
     DO 500 I=1,NELM
          IF(M1.EQ.1.AND.PARAM.EQ.2) THEN
            CALL TRELEM (ELEM,COORD,I,KEL,FEL,VORT,DNDR,DNDS,VI,
     &                        NIPT)
            K1=5
            K3=6
          ELSEIF(M1.EQ.1.AND.PARAM.EQ.3)THEN
            CALL RECTNG (ELEM,COORD,I,KEL,FEL,VORT,SET,DNDR,
     &                        DNDS,VI,NIPT,WI,WJ)
            K1=8
            K3=9
          ELSEIF (M1.EQ.2.AND.(PARAM.EQ.2.OR.PARAM.EQ.3)) THEN
            K1=2
            K3=3
            CALL ELEM2 (ELEM,COORD,I,KEL,FEL)
          ENDIF
          DO 510 LL=1,NPE
            L1=ELEM(I,LL)
            JJ=JDIAG(L1)
            A(JJ)=A(JJ)+KEL(LL,LL)
            A1(JJ)=A1(JJ)+KEL(LL,LL)
            B(L1)=B(L1)+FEL(LL)
            F1(L1)=F1(L1)+FEL(LL)
  510     CONTINUE
          DO 520 L=1,K1
            K2=L+1
            DO 520 J=K2,K3
            M(L,J)=MAX(ELEM(I,L),ELEM(I,J))
            JJ=JDIAG(M(L,J))-ABS(ELEM(I,L)-ELEM(I,J))
            A(JJ)=A(JJ)+KEL(L,J)
            A1(JJ)=A1(JJ)+KEL(L,J)
            C(JJ)=C(JJ)+KEL(J,L)
  520     CONTINUE
  500 CONTINUE
C
C    CONDITION TO FIX THE VELOCITY AT THE SLUICE GATE
C
     IF (M1.EQ.2.AND.EXAM.EQ.3.AND.FLG.NE.1.0) THEN
          JJ=JDIAG(NUM4)
            A(JJ)=A(JJ)+100000000.
          JJ=JDIAG(NUM4+1)
```

```
            A(JJ)=A(JJ)+100000000.
          ENDIF
          RETURN
          END
C
C
          FUNCTION NIMAX(IVEC,NPTS)
                DIMENSION IVEC(3000)
                NIMAX=IVEC(1)
                DO 100 I=2,NPTS
                  NDIF=NIMAX-IVEC(I)
        100       IF(NDIF.LT.0)NIMAX=IVEC(I)
          RETURN
          END
C
C
C ***************************************************************************
C **       SUBROUTINE UACTCL-. SOLVES THE SYSTEM OF EQUATIONS **
C **      KEPT IN A SKYLINE VECTOR. FROM STASA'S BOOK.              **
C ***************************************************************************
C
C
          SUBROUTINE UACTCL(A,C,B,JDIAG,NEQ,AFAC,BACK)
             DIMENSION  A(50000),B(3000),C(50000),JDIAG(3000)
          LOGICAL AFAC,BACK
C
C            FACTOR A TO UT*D*U, REDUCE B TO U
C
          AFAC=.TRUE.
          BACK=.TRUE.
          JR=0
          DO 310 J=1,NEQ
               JD=JDIAG(J)
               JH=JD-JR
               IF(JH.LE.1) GO TO 300
               IS=J+1-JH
               IE=J-1
               IF(.NOT.AFAC) GO TO 250
               K=JR+1
               ID=0
C
C              REDUCE ALL EQUATIONS EXCEPT DIAGONAL
C
              DO 200 I=IS,IE
                IR=ID
                IF(I.EQ.0) GO TO 175
                ID=JDIAG(I)
                 IH=MIN(ID-IR-1,I-IS)
                IF(IH.EQ.0) GO TO 150
                  A(K)=A(K)-DOT(A(K-IH),C(ID-IH),IH)
                  C(K)=C(K)-DOT(C(K-IH),A(ID-IH),IH)
        150       IF(A(ID).EQ.0.0) GO TO 175
                  C(K)=C(K)/A(ID)
        175       K=K+1
```

```
  200   CONTINUE
C
C           REDUCE DIAGONAL TERM
C
          A(JD)=A(JD)-DOT(A(JR+1),C(JR+1),JH-1)
C
C           FORWARD REDUCE THE R.H.S.
C
  250   IF(.NOT.BACK) GO TO 300
        B(J)=B(J)-DOT(C(JR+1),B(IS),JH-1)
  300   JR=JD
  310 CONTINUE
    IF(.NOT.BACK) GO TO 999
C
C           BACK SUBSTITUTION
C
    J=NEQ
    JD=JDIAG(J)
  500 IF(A(JD).EQ.0.0) GO TO 550
      B(J)=B(J)/A(JD)
  550 D=B(J)
    J=J-1
    IF(J.LE.0) GO TO 999
    JR=JDIAG(J)
    IF((JD-JR).LE.1) GO TO 700
    IS=J-JD+JR+2
    K=JR-IS+1
    DO 600 I=IS,J
        B(I)=B(I)-A(I+K)*D
  600 CONTINUE
  700 JD=JR
    GO TO 500
  999 RETURN
    END
C
C           ---------------------------------------------------
C   FUNCTION DOT
C           ---------------------------------------------------
C
    FUNCTION DOT(A,B,N)
      DIMENSION A(50000),B(50000)
    DOT=0.0
    DO 100 I=1,N
        DOT=DOT+A(I)*B(I)
  100 CONTINUE
    RETURN
    END
C
C
C
C ************************************
C *      SUBROUTINE SURF             *
C ************************************
C
```

```
      SUBROUTINE SURF (A1,B,R,NNODE,ELEM,NNY,JDIAG,F1,NEX,NMASX,
     *EL,NELM,NOD,NEL,NPEL,NNX,COORD,COR,NEBCR,NUM4,NUM3,
     *EXAM,NEBG,NMASY,PARAM,VAL3,HEAD,FLG)
          REAL  A1(50000),B(3000),R(300),R1(800,40),COORD(3000,2)
          REAL  R2(800,40),F1(3000),F2(300),COR(3000,2)
         INTEGER  JDIAG(3000),ELEM(1500,9),EL(1500,9),EXAM,NEX(50)
      INTEGER PARAM
     NOD=NNX
      NLX=2*NELM/(NNY-1)
      KONT1=0
C
C  PARAMETERS TO WORK THE SLUICE GATE EXAMPLE
C  NUM1 MAY VARY ACCORDING TO THE NUMBER OF MASTER
C  ELEMENTS ALONG THE SLUICE GATE.
C
       IF (EXAM.EQ.3) THEN
            NOD=0
            DO 5 I=1,NEBG
  5         NOD=NOD+NEX(NMASY*I)-1
            NOD=NOD+1
           NUM2=NOD*NNY
            NUM3=NOD
            DO 6 I=NEBG+1,NEBCR
  6         NUM3=NUM3+NEX(NMASY*I)-1
           NUM3=NUM3*NNY
            DO 7 I=NEBCR+1,NMASX
  7         NOD=NOD+NEX(NMASY*I)-1
            NOD=NOD+1
            NUM4=NUM2/NNY
       ENDIF
C
      KR1=NOD
      KR2=2*NNY+1
       DO 10 I=1,KR1
            DO 20 J=1,KR2
               R1(I,J)=0.
 20         R2(I,J)=0.
 10   R(I)=0.
       KI=0
      K1=NNY
      K2=NNODE
       IF (EXAM.EQ.3) K2=NUM2
      K3=NNY
      KK=0
 15   DO 30 I=K1,K2,K3
            KI=KI+1
            KJ=0
            K4=2*NNY+I
            IF (K4.GT.NNODE)K4=NNODE
            DO 25 J=I,K4
               IF(JDIAG(J)-(J-I).LE.JDIAG(J-1))GO TO 25
              KJ=KJ+1
               R1(KI,KJ)=A1(JDIAG(J)-(J-I))*B(J)
C       WRITE(6,*)'KI,KJ,J,R1,B,JDIAG,A1 =',KI,KJ,J,R1(KI,KJ),B(J),
```

```
C       %JDIAG(J),A1(JDIAG(J)-(J-I))
  25      CONTINUE
  30   F2(KI)=F1(I)
       IF (KK.EQ.0) THEN
            KI2=KI
            KI=0
       ENDIF
       IF (KK.EQ.1) KI=KI2
       DO 40 I=K1,K2,K3
          KI=KI+1
          KJ=0
          L1=JDIAG(I)-1
          L2=JDIAG(I-1)+1
          DO 40 J=L1,L2,-1
            KJ=KJ+1
             R2(KI,KJ)=A1(J)*B(I-KJ)
C         WRITE(6,*)'KI,KJ,J,R2,A1,B(I-KJ)=',KI,KJ,J,R2(KI,KJ),A1(J),
C    & B(I-KJ)
  40  CONTINUE
      KK=KK+1
       IF (EXAM.EQ.3.AND.KONT1.EQ.0) THEN
          K1=NUM3
          K2=NNODE
          KONT1=KONT1+1
          GO TO 15
       ENDIF
       DO 51 I=1,KR1
          DO 51 J=1,KR2
  51   R(I)=R(I)+R1(I,J)+R2(I,J)
C
C    FIX VELOCITY AT THE SLUICE GATE, Vo= Q/Yo
C    ACCORDING TO THE APPROACH VELOCITY IN THE UNIFORM FLOW.
C    ALSO FIX VELOCITY AT TOP NODE WHERE V=0. AND H=HO.
C
       IF (EXAM.EQ.3.AND.FLG.NE.1,0)THEN
          G2=64.4
          V0=VAL3/COORD(NNY,2)
          VEL=SQRT(G2*(HEAD-COORD(NUM3,2)))
          R(NUM4+1)=VEL*100000000.
          R(NUM4)=0.0
       ENDIF
      K1=NNY
      K2=NNODE
       IF (EXAM.EQ.3) K2=NUM2
      K3=NNY
      K4=1
      KK=0
      KT1=0
      K=0
 55   DO 60 I=K1,K2,K3
         K=K+1
         COR(K,1)=COORD(I,1)
 60   COR(K,2)=COORD(I,2)
      IF (EXAM.EQ.3.AND.KONT1.EQ.1) THEN
```

```
          K1=NUM3
          K2=NNODE
          K5=K-1
          KONT1=KONT1+1
          GO TO 55
        ENDIF
        IF (EXAM.NE.3) K5=K-1
        K51=K-1
C
C           THE NEXT "IF" IS FOR CHOOSING LINEAR OR QUADRATIC
C           1-DIMENSIONAL ELEMENTS FOR THE VELOCITIES (PARAM1)
C
      IF (PARAM.EQ.2.OR.PARAM.EQ.3) THEN
  61    DO 62 I=K4,K5,2
          KK=KK+1
          EL(KK,1)=I
          EL(KK,2)=I+1
  62    EL(KK,3)=I+2
        IF (EXAM.EQ.3.AND.KONT1.EQ.2)THEN
          K4=I+1
          K5=K51
          KONT1=KONT1+1
          GO TO 61
        ENDIF
      ELSEIF (PARAM.EQ.1) THEN
  63    DO 64 I=K4,(K5-1)
          KK=KK+1
          EL(KK,1)=I
  64    EL(KK,2)=I+1
        IF(KONT1.EQ.2)THEN
          K4=I+1
          K5=K51
          KONT1=KONT1+1
          GO TO 63
        ENDIF
      ENDIF
      NEL=KK
       IF (PARAM.EQ.2.OR.PARAM.EQ.3) NPEL=3
       IF (PARAM.EQ.1) NPEL=2
      RETURN
      END
C
C
C ***************************************************************************
C **       SUBROUTINE NEWCOR-. CALCULATES THE NEW COORDINATES          **
C **       FOR THE CONTROL POINTS, IE. FOR THE FREE SURFACE            **
C **       MASTER ELEMENTS NODES.                                      **
C ***************************************************************************
C
C
      SUBROUTINE NEWCOR (NX,NMASX,COR,X,Y,R,K,NNY,COORD,VAL2,NEBCR,
     & ERRNOD,EXAM,HEAD,SSS,NEBG,ALFA1,NMASY,NOD,NEX,NUM4,PARAM,
     & AVGFLG,DPNOD,POINT,NPOINT,ITER,SUPX,SUPY,KOUT)
       INTEGER EXAM,NEX(50),NX(50),POINT(101),PARAM,AVGFLG
```

```
      REAL COR(3000,2),X(50,8),Y(50,8),R(300),COORD(3000,2),Q(100)
      REAL DPNOD(75),SUPX(20,5),SUPY(20,5)
      ERR=0.
      ERRNOD=0.
      DPNOD(K)=0.
      NTOT=NOD
      NMAS=NMASX*NMASY
C
C       CREATING THE POINTER ARRAY THAT KEEPS TRACK OF THE CONTROL
C       NODES.
C
      J=NEBG*NMASY
      KT1=NMASY
      KT2=0
      POINT(1)=1
      KK=0
   24 DO 20 I=KT1,J,NMASY
         KK=KK+2
         IF(KT2.EQ.1)THEN
           KK=KK+1
           KT2=KT2+1
         ENDIF
         NUM=(NX(I)*2-2)/2
         DO 20 L=KK,KK+1
           POINT(L)=POINT(L-1)+NUM
   20 CONTINUE
      IF (KT1.EQ.NMASY) THEN
         KT1=NEBCR*NMASY+NMASY
         KT2=KT2+1
         KT3=KK+3
         POINT(KK+2)=POINT(KK+1)+1
         J=NMAS
         GO TO 24
      ENDIF
      NPOINT=L-1
      KK=KK+1
C
C       PROCEDURE TO AVERAGE THE VELOCITIES
C
      IF (EXAM.EQ.3.AND.AVGFLG.EQ.1) THEN
         DO 25 LL=KT3,KK
           L=POINT(LL)
           LO=POINT(LL-1)
           IF(LL.EQ.KK)GO TO 22
           LH=POINT(LL+1)
           D1=COR(L,1)-COR(LO,1)
           D2=COR(LH,1)-COR(L,1)
           Q(L)=(R(LO)*D1+R(LH)*D2+R(L)*(D1+D2))/(2.*(D1+D2))
C          WRITE(6,*)' OLD VELOCITY AT NODE ',L,' = ',R(L),
C    $      ' D1 = ',D1,'  D2 = ',D2
           GO TO 25
   22      CONTINUE
C 22       WRITE(6,*)' OLD VELOCITY AT NODE ',L,' = ',R(L)
           Q(L)=(R(LO)+2.*R(L))/3.
```

```
 25     CONTINUE
        DO 27 L=KT3,KK
 27     R(POINT(L))=Q(POINT(L))
      ENDIF
C
C
C
       V=R(1)**2/2./32.2
      C=HEAD
      J=1-NNY
       IF (K.EQ.ITER.OR.KOUT.EQ.1) WRITE (6,51)
 51   FORMAT (2X,'NODE',4X,'NEW Y',7X,'OLD Y',8X,'X',9X,'VELOCI',
      & 'TY',4X,'ERR',9X,'DP',7X,'DEN',8X,'CORR',5X,'SQRT(2G*(H-Y)')
      KK=0
       ALFA=ALFA1
      KKK=1
      DO 10 I=1,NTOT
          KK=KK+1
          U=R(I)**2/64.4
          DP=COR(I,2)+U-C
          J=J+NNY
C
C
C     LAS SIGUIENTES LINEAS TIENE QUE SER MODIFICADA, AHORITA ES
C     NADA MAS PARA EL EJEMPLO SENCILLO.
C
          IF (COR(I,1).LT.7500.)THEN
             H=(COR(I,2)-(Y(1,1)-ABS((COR(I,1)-X(1,1)))*SSS))**3
             DEN=VAL2**2/32.2/H-1.
            IF(DEN.EQ.0.00)THEN
                WRITE(7,12)  COR(I,1),R(I)
           ELSE
                YCO=COR(I,2)
                COR(I,2)=COR(I,2)+(DP/DEN)*ALFA
            ENDIF
          ENDIF
          ERR=ERR+ABS(YCO-COR(I,2))
 12       FORMAT (/,3X,'X CRITIC = ',F9.3,3X,' NORMAL VEL = ',F9.3)
          VS=64.4*(C-YCO)
          IF (VS.GE.0.0)THEN
            VS=SQRT(VS)
          ELSE
            VS=0.0
          ENDIF
          DPD=DP/DEN
          IF (K.EQ.ITER.OR.PARAM.EQ.3.OR.KOUT.EQ.1) THEN
            WRITE(6,50)I,COR(I,2),YCO,COR(I,1),R(I),ERR,DP,DEN,DPD,VS
          ENDIF
          IF (KK.EQ.POINT(KKK))THEN
            DPNOD(K)=DPNOD(K)+ABS(DP)
          KKK=KKK+1
          ENDIF
 10   CONTINUE
      K1=NMASY
```

```
      K2=NMAS
      IF (EXAM.EQ.3) THEN
          K2=NEBG*NMASY
          KONT2=0
      ENDIF
      KONTN=1
   15 IF(PARAM.EQ.2.OR.PARAM.EQ.3) THEN
          DO 30 I=K1,K2,NMASY
            L=I
             IF (EXAM.EQ.3.AND.KONT2.EQ.1) L=L-(NEBCR-NEBG)
            NUM=NX(I)
            KONTN=KONTN+NUM-1
            NN2=KONTN
             IF (EXAM.EQ.3.AND.KONT2.EQ.1.AND.I.EQ.K1) NN2=NN2+1
            NN1=NN2-NUM+1
            NN3=NN1+NEX(I)-1
           KONTN=NN3
           J=I
            ERRNOD=ERRNOD+ABS(Y(J,4)-COR(NN1,2))
            ERRNOD=ERRNOD+ABS(Y(J,7)-COR(NN2,2))
            ERRNOD=ERRNOD+ABS(Y(J,3)-COR(NN3,2))
            Y(J,4)=COR(NN1,2)
            Y(J,7)=COR(NN2,2)
            Y(J,3)=COR(NN3,2)
             Y(J,6)=(Y(J,3)+Y(J,2))/2.
             X(J,6)=(X(J,3)+X(J,2))/2.
             Y(J,8)=(Y(J,1)+Y(J,4))/2.
             X(J,8)=(X(J,1)+X(J,4))/2.
           KNY=J/NMASY
           SUPY(KNY,4)=Y(J,4)
           SUPY(KNY,3)=Y(J,3)
           SUPY(KNY,5)=Y(J,7)
   30    CONTINUE
     ELSEIF (PARAM.EQ.1) THEN
          DO 40 I=K1,K2,NMASY
            NN2=KONTN+NEX(I)-1
             IF(EXAM.EQ.3.AND.KONT2.EQ.1.AND.I.EQ.K1)NN2=NN2+1
           NN1=NN2-NEX(I)+1
          KONTN=NN2
           WRITE(6,*)'NN1,NN2,I ',NN1,NN2,I
          Y(I,4)=COR(NN1,2)
          Y(I,3)=COR(NN2,2)
         KNY=I/NMASY
          SUPY(KNY,3)=Y(I,3)
          SUPY(KNY,4)=Y(I,4)
           ERRNOD=ERRNOD+ABS(Y(J,4)-COR(NN1,2))
           ERRNOD=ERRNOD+ABS(Y(J,3)-COR(NN3,2))
   40    CONTINUE
     ENDIF
      IF (EXAM.EQ.3.AND.KONT2.EQ.0) THEN
         K1=(NEBCR+1)*NMASY
         K2=NMASX*NMASY
         KONT2=KONT2+1
         GO TO 15
```

```
      ENDIF
   50   FORMAT(2X,I3,4(2X,F10.5),2X,F7.3,4(2X,F9.4))
       WRITE(95,54)
   54  FORMAT(//,6X,'******** NEW ITERATION *******',/)
       DO 55 I=1,NMAS
            IF  (PARAM.EQ.1)WRITE(95,56)  (X(I,J),Y(I,J),J=1,4),2,2
            IF  (PARAM.EQ.2)WRITE(95,57)  (X(I,J),Y(I,J),J=1,8),2,2
            IF  (PARAM.EQ.3)WRITE(95,57)  (X(I,J),Y(I,J),J=1,8),2,2
   55  CONTINUE
    57   FORMAT  (16(F7.4,','),2(I1,','))
    56   FORMAT  (8(F7.4,','),2(I1,','))
   60  RETURN
       END
C
C
C ***************************************************************************
C **        SUBROUTINE OUTMAC-. GENERATES THE FILES THAT ARE              **
C **      USED TO PLOT WITH THE MACINTOSH.                                **
C ***************************************************************************
C
      SUBROUTINE OUTMAC (NNODE,NELM,COOR,ELEM,BB,K,NX,NY,NEBCR,
     & DPNOD,TOL,PARAM,HEAD,VAL3,NNY)
        REAL  COORD(3000,2),B(3000),COOR(3000,2),BB(3000)
       INTEGER  ELEM(1500,9),NX(50),NY(50),PARAM
      KO=K
       IF (PARAM.EQ.3)NNN=222
       IF (PARAM.EQ.2)NNN=220
       IF (PARAM.EQ.1)NNN=210
      IF (DPNOD.LE.TOL)K=99
      WRITE (K,10)
   10  FORMAT(/,3X,'TRANSIENT ANALYSIS = 0',/,3X,'DIMENSION = 2',
     &/,3X,'NUMBER OF VARIABLES = 1',/,3X,'<K> GOVERNING EQUATION',
      &' NUMBERS',/,11X,'1',15X,'1',/,3X,'NUMBER OF PARAMETERS = 0',
     &/,3X,'NUMBER OF BOUNDARY PARAMETERS = 0')
      WRITE (K,20) NNODE,NELM
   20  FORMAT(3X,'NUMBER OF NODES = ',I4,/,3X,'NUMBER OF ELEMENTS =',
     &I5,/,3X,'NUMBER OF BOUNDARY ELEMENTS = 1')
      WRITE (K,30)
    30   FORMAT  (//,4X,'NODE',7X,'X',11X,'Y',11X,'PSI',/)
C
C                  PROCEDURE TO MAKE THE COORD. AND STREAM
C                  FUNCTION UNDIMENSIONAL.
C
          DO 23 I=1,NNODE
             COORD(I,1)=COOR(I,1)/HEAD
             COORD(I,2)=COOR(I,2)/HEAD
   23       B(I)=BB(I)/VAL3
      DO 40 I=1,NNODE
   40   WRITE (K,50) I,COORD(I,1),COORD(I,2),B(I)
   50   FORMAT  (2X,I5,3(2X,F10.5))
      WRITE (K,60)
   60   FORMAT(/,4X,'ELEMENTS')
      DO 70 I=1,NELM
         IF(PARAM.EQ.2)  THEN
```

```
            WRITE (K,80)I,NNN,NNN,ELEM(I,1),ELEM(I,4),ELEM(I,2),
     *      ELEM(I,5),ELEM(I,3),ELEM(I,6)
          ELSEIF(PARAM.EQ.1)THEN
            WRITE(K,85)I,NNN,NNN,ELEM(I,1),ELEM(I,2),ELEM(I,3)
          ELSEIF (PARAM.EQ.3) THEN
            WRITE(K,86)I,NNN,NNN,ELEM(I,1),ELEM(I,5),ELEM(I,2),
     *      ELEM(I,6),ELEM(I,3),ELEM(I,7),ELEM(I,4),ELEM(I,8),
     *      ELEM(I,9)
          ENDIF
   70   CONTINUE
   80   FORMAT (9(2X,I6))
   85   FORMAT (6(2X,I6))
   86   FORMAT (12(2X,I5))
C
C
        IF(PARAM.EQ.2.OR.PARAM.EQ.3)THEN
          N1=NNODE-NNY+1
          N2=(N1-1)/2+1
          WRITE (K,90) 1,N2,N1
        ELSEIF(PARAM.EQ.1)THEN
          WRITE(K,90)ELEM(1,1),ELEM(1,2),ELEM(3,2)
        ENDIF
   90   FORMAT(/,3X,'BOUNDARY ELEMENTS',/,3X,'1   121    121',3(2X,I4),
     &'     0')
        WRITE (K,92) VAL3
   92   FORMAT (/,3X,'DISCHARGE = ',F9.5,' CFS/FT.')
C       CD=VAL3/(
C       CC=COR(
C       Q2=VAL3**2/
        K=KO
        RETURN
        END
C
C
C
C ***************************************************************************
C **      SUBROUTINE SOLVE-. IT SOLVES THE "LUMPED" SYSTEM                 **
C **     OF EQUATIONS GENERATED BY SUBROUTINE SURF, IT WORKS               **
C **     THE VELOCITIES AT THE SURFACE NODES.                              **
C ***************************************************************************
C
C
      SUBROUTINE SOLVE(A,R,JDIAG,NOD,NUM4,EXAM,VAL3,COORD)
        REAL  A(50000),R(300),S1(300,3),S2(300,3),RR(300),
     & COORD(3000,2)
      INTEGER JDIAG(3000),EXAM
C
      KR1=NOD
      DO 10 I=1,KR1
        DO 20 J=1,3
          S1(I,J)=0.
   20     S2(I,J)=0.
   10   RR(I)=0.
        S1(1,1)=A(1)
```

```
      S1(1,2)=A(2)
      S1(1,3)=A(4)
     KI=1
     DO 25 I=2,NOD
          KI=KI+1
          KJ=0
          K4=I+2
          IF (K4.GT.NOD)K4=NOD
         DO 25 J=I,K4
             IF (JDIAG(J)-(J-I).LE.JDIAG(J-1))GO TO 25
            KJ=KJ+1
             S1(KI,KJ)=A(JDIAG(J)-(J-I))
 25  CONTINUE
    KI=1
    DO 40 I=2,NOD
          KI=KI+1
          KJ=0
          L1=JDIAG(I)-1
          L2=JDIAG(I-1)+1
          IF (L2.GT.L1)GO TO 40
          DO 40 J=L1,L2,-1
            KJ=KJ+1
            S2(KI,KJ)=A(J)
 40  CONTINUE
    DO 52 I=1,KR1
         DO 52 J=1,3
 52   RR(I)=RR(I)+S1(I,J)+S2(I,J)
    DO 60 I=1,KR1
 60     R(I)=R(I)/RR(I)
   RETURN
   END
C
C
C ***********************************************************************
C **      SUBROUTINE RELAX-. TO ADJUST THE RELAXATION FACTOR           **
C **      ACCORDING TO THE CONVERGENCE OF THE ITERATIONS.              **
C ***********************************************************************
C
C
    SUBROUTINE RELAX (DPNOD,ALFA1,K,X,Y,YY,NEBCR,NMASY,NMAS,
    &                       NPME,EXAM)
       REAL  DPNOD(75),Y(50,8),YY(50,8),X(50,8)
C
C
     DIF=DPNOD(K-1)-DPNOD(K)
     IF(DIF.GT.0.AND.ALFA1.LT.1.0)THEN
         IF(K.LE.5)RELX=1.05
         IF(K.GT.5.AND.K.LE.10)  RELX=1.075
         IF(K.GT.10.AND.K.LE.15)  RELX=1.10
         IF(K.GT.15.AND.K.LE.20)  RELX=1.125
         IF(K.GT.20)RELX=1.15
         ALFA1=ALFA1*RELX
    ELSEIF (DIF.LE.0.0) THEN
         IF(K.LE.5)RELX=1.075
```

```
          IF(K.GT.5.AND.K.LE.10) RELX=1.13
          IF(K.GT.10.AND.K.LE.15) RELX=1.16
          IF(K.GT.15.AND.K.LE.20) RELX=1.20
          IF(K.GT.20.AND.K.LE.25)RELX=1.25
          IF(K.GT.25)RELX=1.30
          ALFA1=ALFA1/RELX
       ENDIF
         IF (ALFA1.GT.0.15.AND.EXAM.NE.1)ALFA1=0.15
         IF (ALFA1.LT.0.025)ALFA1=0.075
       RETURN
       END
C
C
C ***********************************************************************
C **      SUBROUTINE OUT-. JUST TO PRINT THE RESULTS FOR THE           **
C **      FINAL SOLUTION. IT PRINTS THE FINAL COORDINATES AS           **
C **      WELL AS THE FINAL VELOCITIES AND THE HEAD FOR THE            **
C **      CONTROL NODES.                                               **
C ***********************************************************************
C
C
      SUBROUTINE OUT (COR,R,POINT,NPOINT,K,TOL,DPN,HEAD,DPNOD,
     $                VORFLG,CIRFLG,VORT,KELVOR,KELVC1,KELVC2)
        REAL COR(3000,2),R(300),DPNOD(75),VORT(1500)
      INTEGER POINT(101),VORFLG,CIRFLG
C
    5   FORMAT (//,3X,'FINAL SOLUTION AFTER ',I2,' ITERATIONS WITH',
     &' A TOLERANCE OF SUM.OF DP = ',F7.5,/,7X,'DPN IN LAST ITERA',
     &'TION = ',F9.5,//,10X,'COORDINATES ',
     &'AND VELOCITIES ON THE FREE SURFACE:',//,3X,'NODE',5X,'FINAL',
     &' Y',10X,'X',8X,'VELOCITY',4X,'TOTAL HEAD',2X,'VS=SQRT(2G*(',
     &'H-Y))',/)
   10   FORMAT(3X,I3,5(2X,F10.5))
C
      WRITE (6,5)K,TOL,DPN
      C=HEAD
      DO 20 I=1,NPOINT
          L=POINT(I)
          H=R(L)**2/64.4+COR(L,2)
          VS=64.4*(C-COR(L,2))
          IF (VS.GE.0.0)THEN
             VS=SQRT(VS)
          ELSE
              VS=0.0
          ENDIF
   20   WRITE(6,10)L,COR(L,2),COR(L,1),R(L),H,VS
        WRITE (6,22)
   22   FORMAT(//,3X,'TOTAL DIFFERENCE IN PRESSURE',//,)
        DO 30 I=1,K
   30   WRITE(6,40) I,DPNOD(I)
   40   FORMAT (2X,'DPNOD IN ITERATION # ',I3,' = ',F9.6)
        WRITE (6,42)
   42   FORMAT (//,3X,'VORTICITY WITHIN ELEMENTS',//)
        IF (VORFLG.EQ.1) THEN
```

```
      DO 50 I=1,KELVOR
50    WRITE(6,60) I,VORT(I)
      ENDIF
      IF (CIRFLG.EQ.1) THEN
      DO 55 I=KELVC1,KELVC2
55    WRITE(6,60) I,VORT(I)
      ENDIF
60    FORMAT (3X,'VORTICITY WITHIN ELEMENT # ',I3,' = ',F8.2)
      RETURN
      END
C
C
C*****************************************************************************
C***   SUBROUTINE QCHAN-. THIS SUBROUTINE CHANGES THE VALUE                **
C***   OF THE DISCHARGE UNTIL IT CONVERGES, IT USES A NEWTON               **
C***   RHAPSON'S TYPE SCHEME.                                              **
C*****************************************************************************
C
C
      SUBROUTINE QCHAN (COORD,VAL3,KONTQ,HEAD,ERRQ,VG,NUM3)
C
      REAL COORD(3000,2)
      ALFA=1.0
      VELN2=VG
      VELOC2=SQRT(64.4*(HEAD-COORD(NUM3,2)))
      Q2=VAL3
      IF (KONTQ.EQ.1) THEN
      VELN1=VELN2
      VELOC1=VELOC2
      Q1=VAL3
      F1=VELN1-VELOC1
      VAL3=0.995*VAL3
      GO TO 888
      ELSE
      F2=VELN2-VELOC2
      F1P=(F2-F1)/(Q2-Q1)
      QNEW=Q1-(F1/F1P)*ALFA
      Q1=Q2
      F1=F2
      ERRQ=ABS(VAL3-QNEW)
      VAL3=QNEW
      ENDIF
888   WRITE(6,101)KONTQ
101   FORMAT (//,3X,'ITERATION # ',I2,' FOR THE DISCHARGE')
      KONTQ=KONTQ+1
      WRITE(6,100)Q2,VAL3
100   FORMAT (/,2X,' Q1 = ',F8.5,'   QNEW = ',F8.5)
      RETURN
      END

C
C
C
C*****************************************************************************
```

```
C ** SUBROUTINE DINVOR.- IT COMPUTES THE VORTICITY WITHIN THE        **
C ** ELEMENTS ALONG THE BED AND ALONG THE GATE IF THE FLAG          **
C ** CIRFLG IS OPEN, THE COMPUTATION IS IN AN ELEMENT BASIS.         **
C ***************************************************************
C
C
      SUBROUTINE DINVOR (NNX,NNY,COORD,PSI,PARAM,CIRFLG,NEBG,NEBCR,
     &     NELX,NELY,NELXD,NELYD,VORT,NEX,NMASY,VORIND,VORCIN,KOUT,
     &     KELVOR,KELVC1,KELVC2)
C
       REAL COORD(3000,2),PSI(3000),VORT(1500)
      INTEGER PARAM,CIRFLG,NELX(65),NELY(65),NEX(50)
C
C
     KONT=1
      DO 10 I=1,NNX-1,2
          N1=(I-1)*NNY
         N2=I*NNY
          N3=(I+1)*NNY
           VO1=(PSI(N1+3)-PSI(N1+1))/((COORD(N1+3,2)-COORD(N1+1,2))
     &          **2)
           VO2=(PSI(N2+3)-PSI(N2+1))/((COORD(N2+3,2)-COORD(N2+1,2))
     &          **2)
           VO3=(PSI(N3+3)-PSI(N3+1))/((COORD(N3+3,2)-COORD(N3+1,2))
     &          **2)
          VOT=((VO1+VO2+VO3)/3.)*(-VORIND)
         IF (PARAM.EQ.2) THEN
           VORT(KONT)=VOT
           VORT(KONT+1)=VOT
           KONT=KONT+2
         ELSEIF (PARAM.EQ.3)THEN
           VORT(KONT)=VOT
           KONT=KONT+1
         ENDIF
 10  CONTINUE
     IF (KOUT.EQ.1) THEN
         DO 30 I=1,KONT-1
 30      WRITE(6,35) I,VORT(I)
     ENDIF
 35  FORMAT(3X,'VORTICITY WITHIN ELEMENT # ',I3,' = ',F9.3)
     KELVOR=KONT-1
C
C
     KELVC1=0
     KELVC2=0
     IF (CIRFLG.EQ.1) THEN
         NOD=0
         NUMEL=0
         DO 15 I=1,NEBG
            NOD=NOD+NEX((I-1)*NMASY+1)-1
           NUMEL=NUMEL+NELX((I-1)*NMASY+1)
 15      CONTINUE
         NOD=(NOD+1)*NNY
         NUMEL=NUMEL+(NELYD-1)*NELXD+1
```

```
      IF (PARAM.EQ.2) NUMEL=NUMEL*2-1
      KELVC1=NUMEL
      DO 20 I=NEBG+1,NEBCR
         VO1=(PSI(NOD-2)-PSI(NOD))/((COORD(NOD,1)-COORD(NOD-2,1))
     &          **2)
        NOD=NOD+NNY
         VO2=(PSI(NOD-2)-PSI(NOD))/((COORD(NOD,1)-COORD(NOD-2,1))
     &          **2)
        NOD=NOD+NNY
         VO3=(PSI(NOD-2)-PSI(NOD))/((COORD(NOD,1)-COORD(NOD-2,1))
     &          **2)
         VOT=((VO1+VO2+VO3)/3.)*(-VORCIN)
        IF (PARAM.EQ.2)THEN
         VORT(NUMEL)=VOT
         VORT(NUMEL+1)=VOT
          IF (KOUT.EQ.1) THEN
              WRITE(6,35) (NUMEL),VORT(NUMEL)
              WRITE(6,35) (NUMEL+1),VORT(NUMEL+1)
          ENDIF
         NUMEL=NUMEL+2
        ELSEIF (PARAM.EQ.3) THEN
         VORT(NUMEL)=VOT
          IF (KOUT.EQ.1) WRITE(6,35) (NUMEL),VORT(NUMEL)
         NUMEL=NUMEL+1
        ENDIF
 20    CONTINUE
       IF (PARAM.EQ.2) KELVC2=NUMEL-2
       IF (PARAM.EQ.3) KELVC2=NUMEL-1
   ENDIF
   RETURN
   END
C
C
C
C ***********************************************************************
C **    SUBROUTINE GAUSS.- IT COMPUTES THE VALUES OF THE               **
C **    INTEGRATION POINTS. IT'S USED WITHIN ASEMBL AND THE            **
C **    RESULTS GO INTO THE ELEMENT SUBROUTINES.                       **
C ***********************************************************************
C
C
   SUBROUTINE GAUSS (DNDR,DNDS,VI,PARAM,SET,NIPT,WI,WJ)
C
C
      REAL  DNDR(16,9),DNDS(16,9),VI(16,9),WI(16),WJ(16),R(16)
   REAL S(16)
   INTEGER PARAM,SET
    IF (PARAM.EQ.2)THEN
       R(1)=1./6.
       S(1)=2./3.
       R(2)=2./3.
       S(2)=1./6.
       R(3)=1./6.
       S(3)=1./6.
```

```
      DO 11 I=1,3
         DNDR(I,1)=4.*R(I)+4.*S(I)-3.
         DNDR(I,2)=4.*R(I)-1.
        DNDR(I,3)=0.
         DNDR(I,4)=-8.*R(I)-4.*S(I)+4.
        DNDR(I,5)=4.*S(I)
        DNDR(I,6)=-4.*S(I)
         DNDS(I,1)=4.*S(I)+4.*R(I)-3.
        DNDS(I,2)=0.
         DNDS(I,3)=4.*S(I)-1.
        DNDS(I,4)=-4.*R(I)
        DNDS(I,5)=4.*R(I)
         DNDS(I,6)=-8.*S(I)-4.*R(I)+4.
          VI(I,1)=(1.-R(I)-S(I))*(2.*(1.-R(I)-S(I))-1.)
        VI(I,2)=(2.*R(I)*R(I)-R(I))
        VI(I,3)=(2.*S(I)*S(I)-S(I))
         VI(I,4)=4.*R(I)*(1.-R(I)-S(I))
       VI(I,5)=4.*R(I)*S(I)
         VI(I,6)=4*S(I)*(1.-R(I)-S(I))
       NIPT=3
11    CONTINUE
   ELSEIF (PARAM.EQ.3) THEN
      IF (SET.EQ.1) THEN
        DO 12 I=1,4
          WI(I)=1.
          WJ(I)=1.
            R(I)=0.577350269189626
         S(I)=R(I)
12      CONTINUE
        R(2)=(-R(1))
       R(4)=R(2)
       S(1)=R(2)
       S(2)=S(1)
      NIPT=4
     ELSEIF (SET.EQ.2) THEN
       DO 13 I=1,6
         WI(I)=1.
           WJ(I)=0.555555555555556
13      CONTINUE
         WJ(3)=0.888888888888889
        WJ(4)=WJ(3)
         R(1)=0.577350269189626
       R(2)=(-R(1))
      R(3)=R(1)
      R(4)=R(2)
      R(5)=R(1)
      R(6)=R(2)
         S(1)=(-0.77459669241483)
      S(2)=S(1)
      S(3)=0.
      S(4)=0.
       S(5)=(-S(5))
      S(6)=S(5)
     NIPT=6
```

```
      ELSEIF (SET.EQ.3)THEN
        DO 14 I=1,3
             WI(I)=0.555555555555556
          WJ(I)=WI(I)
14      CONTINUE
        DO 15 I=4,6
             WI(I)=0.888888888888889
          WJ(I)=WI(I)
15      CONTINUE
        DO 16 I=7,9
          WI(I)=WI(1)
          WJ(I)=WI(I)
16      CONTINUE
        DO 17 I=1,7,3
17        R(I)=0.77459669241483
        DO 18 I=2,8,3
18      R(I)=0.
        DO 19 I=3,9,3
19      R(I)=(-R(1))
        DO 20 I=1,3
20      S(I)=(-R(1))
        DO 21 I=4,6
21      S(I)=0.
        DO 22 I=7,9
22      S(I)=R(1)
        NIPT=9
      ELSEIF (SET.EQ.4) THEN
        DO 31 I=1,4
             WJ(I)=0.347854845137454
          WJ(I+12)=WJ(I)
             S(I)=(-0.861136311594053)
31      S(I+12)=(-S(I))
        DO 32 I=5,8
             WJ(I)=0.652145154862546
          WJ(I+4)=WJ(I)
             S(I+4)=0.339981043584856
32      S(I)=(-S(I+4))
        DO 33 I=1,13,4
          WI(I)=WJ(1)
          WI(I+3)=WJ(1)
          R(I)=S(13)
33      R(I+3)=S(1)
        DO 34 I=2,14,4
          WI(I)=WJ(6)
          WI(I+1)=WJ(6)
          R(I)=S(10)
34      R(I+1)=S(6)
        NIPT=16
      ENDIF
C
      DO 25 I=1,NIPT
        T=0.25
        TR=2.*R(I)
        TS=2.*S(I)
```

```
            MR=1.-R(I)
            MS=1.-S(I)
            PR=1.+R(I)
            PS=1.+S(I)
           PMR=PR*MR
           PMS=PS*MS
C
            VI(I,3)=T*PR*PS*R(I)*S(I)
            VI(I,2)=(-T*PR*MS*R(I)*S(I))
           VI(I,1)=T*R(I)*MR*S(I)*MS
            VI(I,4)=(-T*R(I)*MR*S(I)*PS)
            VI(I,6)=.5*R(I)*PR*PMS
            VI(I,5)=(-.5*PMR*S(I)*MS)
            VI(I,8)=(-.5*R(I)*MR*PMS)
            VI(I,7)=.5*PMR*S(I)*PS
           VI(I,9)=PMR*PMS
C
            DNDR(I,3)=T*S(I)*PS*(1.+TR)
            DNDR(I,2)=(-T*S(I)*MS*(TR+1.))
           DNDR(I,1)=T*S(I)*MS*(1.-TR)
            DNDR(I,4)=(-T*S(I)*PS*(1.-TR))
           DNDR(I,6)=.5*PMS*(1.+TR)
          DNDR(I,5)=R(I)*S(I)*MS
           DNDR(I,8)=(-.5*(1.-TR)*PMS)
           DNDR(I,7)=(-R(I)*S(I)*PS)
           DNDR(I,9)=(-2.*R(I)*PMS)
C
          DNDS(I,3)=T*R(I)*PR*(1.+TS)
           DNDS(I,2)=(-T*R(I)*PR*(1.-TS))
          DNDS(I,1)=T*R(I)*MR*(1.-TS)
          DNDS(I,4)=(-T*R(I)*MR*(1.+TS))
           DNDS(I,6)=(-S(I)*R(I)*PR)
           DNDS(I,5)=(-.5*PMR*(1.-TS))
         DNDS(I,8)=S(I)*R(I)*MR
          DNDS(I,7)=.5*PMR*(1.+TS)
          DNDS(I,9)=(-2.*S(I)*PMR)
 25     CONTINUE
    ENDIF
   RETURN
   END
C
C
C ***********************************************************************
C ** SUBROUTINE DIS.- TO COMPUTE THE NUMERICAL VELOCITY AT              **
C ** THE TIP OF THE GATE, FROM HERE IT GOES TO 'QCHAN' WHERE            **
C ** THE DISCHARGE IS ITERATED.                                         **
C ***********************************************************************
C
C
   SUBROUTINE DIS (NUM3,NNY,COORD,DNDR,DNDS,B,VG)
C
      REAL  COORD(3000,2),DNDR(16,9),DNDS(16,9),PS(3,6),J11B(3,3),
   %   J12B(3,3),J21B(3,3),J22B(3,3),XX(50,8),YY(50,8),AA(3,3,6),
   %        BB(3,3,6),DETT(3,3),VV(3,3),UU(3,3),B(3000),CCX(3,3),
```

```
%       CCY(3,3),RC(3),SC(3),VELL(3,3)
      EXTERNAL SHAPE
C
       RC(1)=1./6.
       RC(2)=2./3.
       RC(3)=1./6.
       SC(1)=2./3.
       SC(2)=1./6.
       SC(3)=1./6.
C
C
      XX(1,1)=COORD(NUM3,1)
      YY(1,1)=COORD(NUM3,2)
      PS(1,1)=B(NUM3)
      XX(1,2)=COORD(NUM3-2*NNY,1)
      YY(1,2)=COORD(NUM3-2*NNY,2)
      PS(1,2)=B(NUM3-2*NNY)
      XX(1,3)=COORD(NUM3-2,1)
      YY(1,3)=COORD(NUM3-2,2)
      PS(1,3)=B(NUM3-2)
      XX(1,4)=COORD(NUM3-NNY,1)
      YY(1,4)=COORD(NUM3-NNY,2)
      PS(1,4)=B(NUM3-NNY)
      XX(1,5)=COORD(NUM3-2*NNY-1,1)
      YY(1,5)=COORD(NUM3-2*NNY-1,2)
      PS(1,5)=B(NUM3-2*NNY-1)
      XX(1,6)=COORD(NUM3-1,1)
      YY(1,6)=COORD(NUM3-1,2)
      PS(1,6)=B(NUM3-1)
C
C
      XX(2,1)=COORD(NUM3-2,1)
      YY(2,1)=COORD(NUM3-2,2)
      PS(2,1)=B(NUM3-2)
      XX(2,2)=COORD(NUM3+2*NNY-2,1)
      YY(2,2)=COORD(NUM3+2*NNY-2,2)
      PS(2,2)=B(NUM3+2*NNY-2)
      XX(2,3)=COORD(NUM3,1)
      YY(2,3)=COORD(NUM3,2)
      PS(2,3)=B(NUM3)
      XX(2,4)=COORD(NUM3+NNY-2,1)
      YY(2,4)=COORD(NUM3+NNY-1,2)
      PS(2,4)=B(NUM3+NNY-2)
      XX(2,5)=COORD(NUM3+NNY-1,1)
      YY(2,5)=COORD(NUM3+NNY-1,2)
      PS(2,5)=B(NUM3+NNY-1)
      XX(2,6)=COORD(NUM3-1,1)
      YY(2,6)=COORD(NUM3-1,2)
      PS(2,6)=B(NUM3-1)
C
C
     XX(3,1)=COORD(NUM3+2*NNY,1)
     YY(3,1)=COORD(NUM3+2*NNY,2)
     PS(3,1)=B(NUM3+2*NNY)
```

```
      XX(3,2)=COORD(NUM3,1)
      YY(3,2)=COORD(NUM3,2)
      PS(3,2)=B(NUM3)
      XX(3,3)=COORD(NUM3+2*NNY-2,1)
      YY(3,3)=COORD(NUM3+2*NNY-2,2)
      PS(3,3)=B(NUM3+2*NNY-2)
      XX(3,4)=COORD(NUM3+NNY,1)
      YY(3,4)=COORD(NUM3+NNY,2)
      PS(3,4)=B(NUM3+NNY)
      XX(3,5)=COORD(NUM3+NNY-1,1)
      YY(3,5)=COORD(NUM3+NNY-1,2)
      PS(3,5)=B(NUM3+NNY-1)
      XX(3,6)=COORD(NUM3+2*NNY-1,1)
      YY(3,6)=COORD(NUM3+2*NNY-1,2)
      PS(3,6)=B(NUM3+2*NNY-1)
C
C
      DO 12 K=1,3
         DO 12 I=1,3
            J11B(K,I)=0.
            J12B(K,I)=0.
            J21B(K,I)=0.
            J22B(K,I)=0.
           UU(K,I)=0.
           VV(K,I)=0.
           VELL(K,I)=0.
   12 CONTINUE
      DO 13 K=1,3
         DO 13 I=1,3
            DO 13 J=1,6
               J11B(K,I)=J11B(K,I)+DNDR(I,J)*XX(K,J)
               J12B(K,I)=J12B(K,I)+DNDR(I,J)*YY(K,J)
               J21B(K,I)=J21B(K,I)+DNDS(I,J)*XX(K,J)
               J22B(K,I)=J22B(K,I)+DNDS(I,J)*YY(K,J)
   13 CONTINUE
      DO 20 K=1,3
         DO 20 I=1,3
            DETT(K,I)=(J11B(K,I)*J22B(K,I))-(J21B(K,I)*J12B(K,I))
   20 CONTINUE
      DO 30 K=1,3
         DO 30 I=1,3
            DO 30 J=1,6
        AA(K,I,J)=(J22B(K,I)*DNDR(I,J)-J12B(K,I)*DNDS(I,J))/DETT(K,I)
        BB(K,I,J)=(J11B(K,I)*DNDS(I,J)-J21B(K,I)*DNDR(I,J))/DETT(K,I)
   30 CONTINUE
      DO 50 K=1,3
         DO 41 I=1,3
            DO 40 J=1,6
               VV(K,I)=VV(K,I)+AA(K,I,J)*PS(K,J)
               UU(K,I)=UU(K,I)+BB(K,I,J)*PS(K,J)
   40       CONTINUE
            VELL(K,I)=SQRT(VV(K,I)**2+UU(K,I)**2)
   41    CONTINUE
   50 CONTINUE
```

```
      DO 70 K=1,3
         DO 70 I=1,3
           CCX(K,I)=SHAPE(RC(I),SC(I),XX,K,5)
           CCY(K,I)=SHAPE(RC(I),SC(I),YY,K,5)
70    CONTINUE
      D1=SQRT((XX(1,1)-CCX(1,3))**2+(YY(1,1)-CCY(1,3))**2)
      D2=SQRT((XX(1,1)-CCX(2,1))**2+(YY(1,1)-CCY(2,1))**2)
      D3=SQRT((XX(1,1)-CCX(3,2))**2+(YY(1,1)-CCY(3,2))**2)
       VG=(VELL(1,3)*D1+VELL(2,1)*D2+VELL(3,2)*D3)/(D1+D2+D3)
     WRITE (6,*) 'VELOCITY AT THE GATE = ',VG
     RETURN
     END
```